ANDROID BASED APPLICATION TO LOCATE USEFUL PLACES

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Mayank Tiwari (141205)
Vineesh Singh (141226)

Under the supervision of

Mr.Amol Vasudeva (Assistant Professor (grade II))

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Android based application to locate useful places "** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2016 to December 2016 under the supervision of **Mr. Amol Vasudeva** (Assistant Professor (grade II), CSE).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Mayank Tiwari (141205)

Vineesh Singh (141226)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Mr. Amol Vasudeva

(Assistant Professor (grade II))

CSE

Dated:

ACKNOWLEDGEMENT

I should need to express my most huge gratefulness to every last one of the general population who gave me the likelihood to finish this report. A unique gratefulness I oblige our last year meander manage, **Mr. Amol Vasudeva,** whose devotion in empowering recommendation and support, helped me to energize my meander particularly in framing this report and besides have put his full exertion in controlling the social occasion in accomplishing the objective. I need to regard the heading given by other official and furthermore the sheets particularly in our undertaking introduction that has enhanced our introduction aptitudes because of their remark and advices.

Other than I may in like way need to see with much appreciation the fundamental bit of the staff of Computer Science department,who gave the consent to utilize all required hardware and the essential materials to finish the undertaking.

TABLE OF CONTENT

LIST OF ABBREVIATIONS

ACRONYM	DEFINITIONS
1. SQL	System Query Language
2. AAC	Advance Audio Coding
3. AMR	Adapter Multi Rate
4. JPEG	Joint Photography experts group
5. SGL	Scalable Graphic Library
6. SDK	Software development kit
7. APK	Android package kit
8. UML	Unified modeling language
9. QOS	Quality of Service
10. API	Application program Interface
11. IPC	Inter Process Communication
12. UI	User Interface
13. ANR	Application not Responding
14. GLIBC	GNU C Library
15. JDT	Java Development Tool

LIST OF FIGURE

LIST OF TABLES

ABSTRACT

Smartphone applications have found their way into almost every avenue in our lives. This project aims to develop a mobile application based on location based services As the efficiency and success of an Internet is increased, this project report emphasizes on how our mobile application can effectively contribute in finding nearby places and contact information and route.

Chapter -1 INTRODUCTION

1.1 Introduction

Android is an adaptable working structure (OS) beginning at now made by Google, in context of the Linux Kernel and shaped basically for touch screen cell phones, for example, PDAs and tablets. Android's UI is by and large in light of direct control, utilizing touch signals that wholeheartedly relate to real activities, for example, swiping, tapping and squeezing, to control on-screen objects, near to a virtual for content information. Regardless of touch screen gadgets, Google has moreover made Android TV for TVs, Android Auto for autos, and Android Wear for wrist watches, each with a particular UI. Assortments of Android are in like way utilized on scratch pad, distraction comfort, motorized cameras, and differing gear.

Android Components-Application system empowering reuse and substitution of pieces Dalvik virtual machine improved for cell phones Integrated program in context of the open source Web Kit motor Optimized portrayals invigorated by a custom 2D designs library; 3D traces in light of the OpenGL ES particular (equip fortifying discretionary)

SQLite for dealt with information gathering

Media strengthen for principal sound, video, and still picture designs (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
GSM Telephony (equipment subordinate)

Bluetooth, EDGE, 3G, and Wi-Fi (equipment subordinate)

Camera, GPS, compass, and accelerometer (equipment subordinate)

Google Maps: Whether scanning for the ideal eatery, looking best lodgings or finding the closest bank, innumerable around the globe get Google Maps to do the vigorous work for them. So for what reason not do in like way for your own particular site? The Google

Maps API is one of those cunning bits of Google progression that causes you take the essentialness of Google Maps and put it coordinate disconnected site. It enables you to fuse pertinent substance that is strong to your guests and change site look and feel of the manual for fit with the style of your site. With more than 150,000 objectives beginning at now utilizing the Google Maps API, we couldn't fit them all into this booklet so we picked a few the most beneficial and inventive cases to help move you. Additionally, if after that you're 'as of recently hungry for additional, look at the back of this booklet for relationship with more outlines and particular data. Google maps is an extraordinary procedure for examine the zone around the property that you are had with, sparing you hours of time and dissatisfaction being exhibited properties that don't sort out your pursuit criteria. Google maps, together with diagram the full video of the property, fills in as a practical instrument when short posting properties that you wish to physically watch. With Google maps you will be able to climb and not far-removed similarly as you were strolling around it. You can see satellite perspectives of the property and wrapping zone, see a guide, get headings and GPS direction to the property and even demand abutting working environments, for example, schools, spots of affection and strip mall. As you wind up being more comfortable with Google maps you will find that there are unmistakable approaches to manage play out specific undertakings. You will in like way find a few solutions concerning different particular highlights that Google maps passes on to the table, that are not all canvassed in this report. Google Maps is a work zone web mapping association made by Google. It offers satellite symbolism, guides, 360° far reaching perspectives of streets (road View) constant activity conditions (Google improvement) , and course making amusement arrangements for going by foot, auto, bike , or transportation.

Google Maps' satellite view is a "best down" view; a gigantic piece of the high-affirmation symbolism of urban locales is airborne photography taken from plane flying at 800 to 1,500 feet (240 to 460 m), while most other symbolism is from satellites. A marvelous bit of the accessible satellite symbolism is almost three years of age and is stimulated continually. Google Maps utilizes a nearby assortment of the Mercator projection, and along these lines can't definitively indicate zones around the posts.

The current revived translation of the work locale application was made open in 2013, contiguous the "centerpiece" (pre-2013) outline. Google Maps for flexible was discharged in September 2008 and highlights GPS turn by turn course. In August 2013, it was embarked to be the world's most remarkable application for bleeding edge cells , with more than 54% of general PDA proprietors utilizing it in any event once.

1.2 Problem Statement

Currently there are android apps in the market which will help you to book hotel rooms and apps that track your locations but no single app with both these properties. So we are developing this app with both these features (finding tourist spots and hotels near your location and tracking you moreover its reviews and gallery to view images).

1.3 Objectives

1. To help tourists to search and find tourist spots near their location .
2. To reduce their search time and problem of visiting one websites/app to find tourist spots and other website/app for booking.
3. Providing best prices in hotels (linkages, discounts, coupons).
4. Light app which doesn't consumes much net.
5. Track the user location and help him to reach the desired location.

1.4 Methodology

The present programming things are mind boggling and reliably developing all through the whole thing cycle. That makes it harder than at later to hold everybody in comprehension.

Handy procedures decrease overhead, however increment chance by expelling process steps which are fundamental to regulating more noteworthy assignments. The correct response is to not surrender time-endeavored techniques and annals. The reasonable

response is to make the creation and support of programming building reports profoundly speedier and more viable.

The Agile framework is as regularly as conceivable utilized by new associations for new Android minimal plans where the adaptable pros require speedy input. It's in addition used by IT affiliations whose inside clients can't surrender to what they require.

At Android Developers Ltd we utilize deft movement comprehensively crosswise over completed Android helpful change assignments of various scale and needs.

Dexterous programming progress thinks about quick turns and making necessities.

Programming building is the exhibit of utilizing picked process methodologies to refresh the possibility of a thing change exertion. This depends upon the uncertainty, subject to unending down to earth talk and upheld by tireless experience, that a profitable strategy to oversee programming change prompts less distortions and, therefore, over the long haul gives shorter development times and better respect.

Two procedures which have displayed to work when necessities are not in all likelihood knew are the Rational Unified Process and Agile Software Development.

Purposes of enthusiasm with Agile Software Development:

1. Obliged meander shot

2. Broadened meander distinguishable quality

3. Upgraded consistency, and versatility

4. Time of all things considered splendid programming

5. Accessibility of different choices to track and survey the undertaking a significant part of the time

6. Cost Control

Most deft programming approachs lay feature on building releasable programming in a word periods essentially like other iterative change models. Deft movement shifts from other change models in two key viewpoints. These fundamental complexities are the short and strict circumstances for each highlight, every last one of which don't beat week, and an essentially accommodating strategy for working with the customer.

1.5 Organization

Chapter 1: Highlights and Underlines of the Location based administrations. In this section, the presentation Location based administrations is secured. The key concentration characterizing the issue articulation and determining the destinations of the task .

Chapter 2: The nitty gritty writing audit from the exploration paper, books, diaries and gatherings are finished. In this part, the concentrates from grouped research papers on HCI, Location Tracking, Tourism.

Chapter 3: Covers the framework improvement which is the key part of this work. In this part, the proposed demonstrate, calculation, UML charts and related parameters are accentuated.

Chapter 4: The reenactment of usage comes about with the relative execution investigation is appeared in this part. The reenactment results and screen captures are uncovered to delineate and protect the proposed work.

Chapter 5: Detailed conclusion and extent without bounds work which directs the forthcoming understudies and research researchers to improve the ebb and flow work with higher proficiency and adequacy on Location racking and tourism.

Chapter -2 LITERATURE SURVEY

2.1 Literature Survey

Prior, handheld GPS beneficiaries, for example, a Garmin Etrex, were being used for various years and have been utilized to update understudy learning through geo securing and adaptable mapping. With change, 10 years back has seen another capacity to geo name photos by utilizing two separate gadgets: a GPS gatherer and a camera. These gadgets are related when and date settings on every last one of the contraptions and a brief timeframe later an outer application was utilized to synchronize them.

Before long front line phones have both mechanized cameras and helped GPS in-endeavored to the gadget. The relative straightforwardness and openness of geo checking has "made a union of geo-mind"

US News and World Report records geotagging photographs as one of "50 approaches to manage update your life in 2009". Friedland and Sommer express that "all the tremendous moved cell producers are before long offering models permitting brief trade of geotagged photographs, accounts, and even messages to territories such Flickr, YouTube, and Twitter."

In a current overall survey of topography and bioscience moved direction experts drove by the creators, geotagging was more than once refered to as one zone of advancement that pros might want to see connecting all through the going with five years, a slant shared by Luo et al. who prescribe that, "with the transparency of web, GPS contraptions and moved mobile phones, the expansion and accessibility of geotagged media will keep expanding". Moreover, Johnson et al. (2010) see the utilization of telephones in rule as one of the key zones in which they expect fundamental change in next a year.

In setting with the exactness, there have been recommendation in enhancing the Quality of Service (QoS) utilizing crowdsourcing . In like way relative examination concerning exactness of region information has been done among Android and iOS .

2.2 Android Architecture

Android architecture has mainly 4 following layers:

1. Applications Layer
2. Application Framework
3. Libraries along with android runtime libraries
4. Linux Kernel.

The following diagram shows the architecture in proper stack

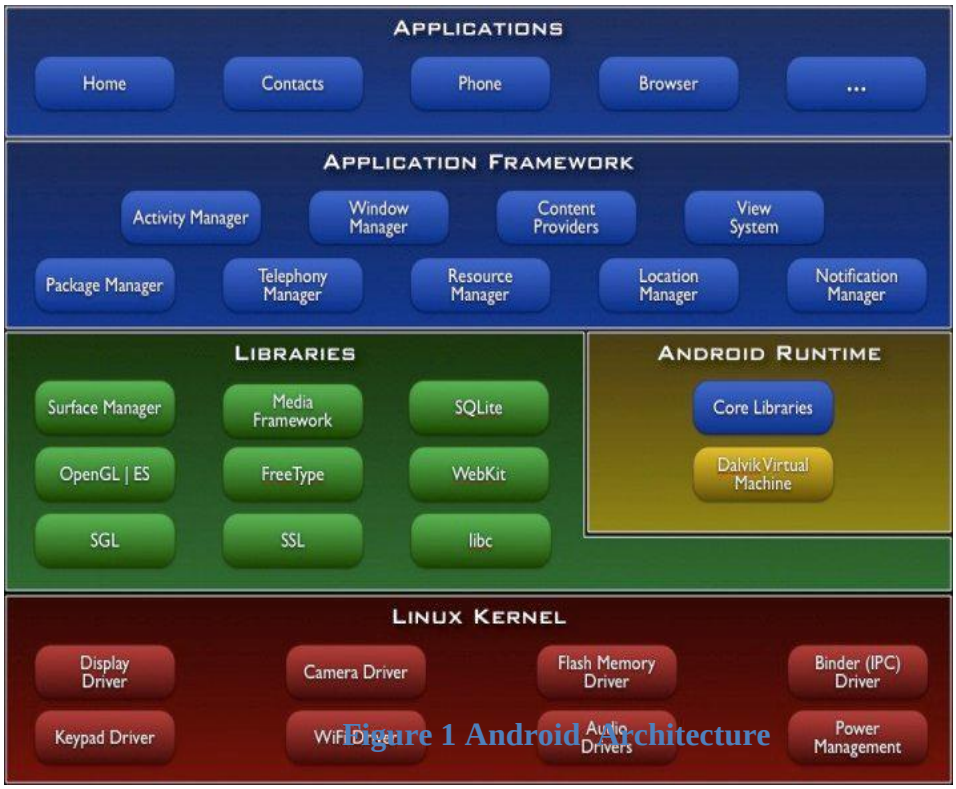


Figure 1 Android Architecture

Applications Layer

Android will convey with a course of action of focus applications including an email client, SMS program,

logbook, maps, program, contacts, and others. All applications are formed using the Java programming vernacular. These applications incorporate the application layer of android.

Application Framework

Application structure gives assembles full access to a tantamount system APIs utilized by the center applications. The application outlining is wanted to improve the reuse of sections; any application can proper its capacities and some other application may then make utilization of those limits (subject to security imperatives kept up by the structure). This same structure engages parts to be supplanted by the client.

Central all applications is a game-plan of associations and frameworks, including:

A rich and extensible strategy of Views that can be utilized to make an application, including records, grids, content boxes, gets, and even an embeddable web program.

Content Providers that draw in applications to get to information from different applications, (for example, Contacts), or to share their own specific information.

A Resource Manager, offering access to non-code assets, for example, restricted strings, layouts, and arrangement chronicles.

A Notification Manager that connects with all applications to show custom alerts in the status bar.

An Activity Manager that game plans with the lifecycle of occupations and gives a normal course backstack.

Libraries

Android intertwines a strategy of C/C++ libraries utilized by different parts of the Android structure. These limits are displayed to engineers through the Android application structure. A section of within libraries are recorded underneath:

Structure C library - a BSD-inferred execution of the standard C framework library (libc), tuned for installed Linux-based contraptions

Media Libraries - Based on Packet Video's Open CORE; the libraries bolster playback and recording of different standard sound and video social occasions, and besides static picture files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG.

Surface Manager - guides access to the show subsystem and dependably composites 2D and 3D down to earth layers from various applications

LibWebCore - a cutting edge web program motor which powers both the Android program and an embeddable web see

SGL - the covered 2D diagrams motor

3D libraries - an utilization in light of OpenGL ES 1.0 APIs; the libraries utilize either outfit 3D restoring (where accessible) or the included, especially upgraded 3D programming rasterizer

FreeType - bitmap and vector printed style rendering

SQLite - A skillful and lightweight social database motor open to all applications

Android Runtime Libraries

Android joins a course of action of focus libraries that gives most of the handiness available in the middle libraries of the Java programming lingo.

Every Android application continues running in its own specific technique, with its own specific event of the Dalvik virtual machine. Dalvik has been surrounded with the objective that a contraption can run distinctive VMs viably. The Dalvik VM executes records in the Dalvik Executable (.dex) deal with which is streamlined for insignificant memory impression. The VM is enroll based, and runs classes accumulated by a Java dialect compiler that have been changed into the .dex make by the included "dx" contraption.

The Dalvik VM relies on the Linux part to basic handiness, for instance, threading and low-level memory affiliation.

Linux Kernel

Android relies on Linux assortment 2.6 for focus structure relationship, for instance, security, memory affiliation, process affiliation, facilitate stack, and driver show up. The piece similarly goes about as a reflection layer between the gear and the straggling stays of the thing stack.

2.3 Main Components of Android Application

There are 5 components around which an android applications revolves. They are

1. Activities
2. Broadcast Receivers
3. Services
4. Intents
5. Content Providers

Activities

An Activity is an application part that outfits a screen in light of which customers can coordinate keeping the true objective to achieve something, for instance, dial the phone, take a photo, send an email, or view a guide. Each development is given a window in which to draw its UI. The window consistently fills the screen, however may be more diminutive than the screen and float over various windows.

An application as a rule involves various activities that are around bound to each other. Typically, one activity in an application is resolved as the "essential" activity, which is shown to the customer while driving the application unexpectedly. Each development would then have the capacity to start another activity with a particular ultimate objective to perform unmistakable exercises. Each time another development starts, the past activity is stopped, however the system ensures the activity in a stack (the "back stack"). Exactly when another development starts, it is pushed onto the back stack and takes customer focus. The back stack endures to the basic "rearward in, first out" line instrument, in this way, when the customer is done with the present activity and presses the BACK key, it is flown from the stack (and pounded) and the past development resumes. (The back stack is analyzed more in the Tasks and Back Stack report.)

Right when a development is ended because another activity starts, it is recounted this alteration in state through the activity's lifecycle callback strategies. There are a couple of callback techniques that a development may get, as a result of a modification in its state—paying little mind to whether the system is making it, stopping it, proceeding with it, or crushing it—and each callback allows you to perform specific work that is appropriate to that state change. For instance, when stopped, your development should release any significant things, for instance, framework or database affiliations. Exactly when the activity resumes, you can reacquire the principal resources and resume exercises that were meddled. These state progresses are all bit of the development lifecycle.

Broadcast Receivers

Broadcast Receiver is extremely a framework to send and get events so all fascinated applications can be taught when something happens. There are heaps of System events which get convey by Android OS, for instance, SMS related events, Connectivity related events, and camera related events and some more. We can convey our application specific events as well. It would be a shrewd idea to use Broadcast Receiver procedure, not simply in light of the fact that it will separate your event dealing with code however specifically since it will enable diverse applications to enlist and get a notice at whatever point that event happens.

There are two important classes of conveys that can be gotten:

Normal Broadcast are absolutely unique. All gatherers of the convey are continue running in a dubious demand, much of the time meanwhile. This is more compelling, however suggests that beneficiaries can't use the result or rashly end APIs included here.

Ordered Broadcast are passed on to one gatherer at any given minute. As each recipient executes hence, it can multiply a result to the accompanying authority, or it can absolutely rashly end the speak with the objective that it won't be passed to various beneficiaries. The ask for authorities continue running in can be controlled with the need property of the organizing objective channel; recipients with a comparative need will be continue running in a self-order.

Services

A Service is an application segment that can perform long-running activities out of sight and does not give a UI. Another application part can begin an administration and it will keep on running out of sight regardless of whether the client changes to another application. Moreover, a segment can tie to an administration to connect with it and even perform between process correspondence (IPC). For instance, an administration may deal with arrange exchanges, play music, perform record I/O, or connect with a substance supplier, all from the foundation.

Bound

An administration is "bound" when an application segment ties to it by calling bindService(). A bound administration offers a customer server interface that enables parts to associate with the administration, send demands, get comes about, and even do as such crosswise over procedures with between process correspondence (IPC). A bound administration runs just as long as another application part is bound to it. Various parts can tie to the administration without a moment's delay, however when every one of them unbind, the administration is decimated.

Content Providers

Content suppliers store and recover information and make it available to all applications. They're the best way to share information crosswise over applications; there's no regular stockpiling territory that all Android bundles can get to.

Android accompanies various substance suppliers for normal information writes (sound, video, pictures, individual contact data, et cetera). You can see some of them recorded in the android. supplier bundle. You can inquiry these suppliers for the information they contain (despite the fact that, for a few, you should secure the best possible consent to peruse the information).

In the event that you need to make your own information open, you have two choices: You can make your own particular substance supplier (a Content Provider subclass) or you can add the information to a current supplier — if one controls a similar sort of information and you have consent to keep in touch with it.

Intent

Three of the center parts of an application — exercises, administrations, and communicate beneficiaries — are actuated through messages, called plans. Expectation informing is an office for late run-time official between parts in the same or diverse applications. The goal itself, an Intent protest, is a latent information structure holding a conceptual portrayal of a task to be performed — or, regularly on account of communicates, a depiction of something that has happened and is being declared. There are separate instruments for conveying purposes to each kind of segment:

An Intent protest is passed to Context startActivity() or Activity startActivityForResult() to dispatch a movement or get a current action to do something new.(It can likewise be passed to Activity setResult() to return data to the action that called startActivityForResult()).

An Intent question is passed to Context startService() to start an administration or convey new guidelines to a progressing administration. Thus, an aim can be passed to setting bindService() to build up an association between the calling part and an objective administration. It can alternatively start the administration if it's not effectively running.

For each situation, the Android framework finds the proper action, administration, or set of communicate beneficiaries to react to the plan, instantiating them if important. There is no cover inside these informing frameworks: Broadcast purposes are conveyed just to communicate recipients, never to exercises or administrations. A plan go to startActivity() is conveyed just to an action, never to a service or broadcast

Activity Stack

• Activities in the framework y are overseen as a movement stack.

• When another action is begun, it is put on the highest point of the stack and turns into the running movement -- the past action dependably stays beneath it in the stack, and won't go to the forefront again until the point when the new action exits.

• If the client presses the Back Button the following action on the stack climbs and winds up dynamic.

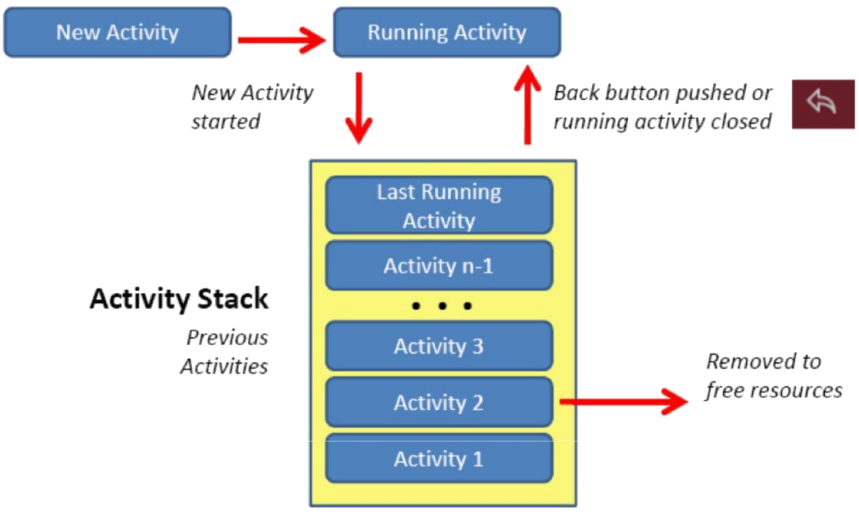


Figure 2 Activity Layout

2.1.6 Processes and Threads

At the point when an applicationpart begins and the application does not have some other segments running, the Android framework begins another Linux procedure for the application with a solitary string of execution. As a matter of course, all segments of a similar application keep running in a similar procedure and string (called the "principle" string). On the off chance that an application segment begins andthere as of now exists a procedure for that application (on the grounds that another part from the application exists), at that point the segment is begun inside that procedure and utilizations a similar string of execution. Be that as it may, you can mastermind distinctive segments in your applicationto keep running in independent procedures, and youcanmake extra strings for any procedure.

When choosing which procedures to slaughter, the Android framework measures their relative significance to the client. For instance, it all the more promptly close down a procedure facilitating exercises that are never again obvious on screen, contrasted with a procedure facilitating unmistakable exercises. The choice whether to end a procedure, in this manner, relies upon the condition of the segments running in that procedure.

At the point when an applicationis propelled, the framework makes a string of execution for the application, called "fundamental." This string is critical in light of the fact that it is responsible for dispatching occasions to the suitable UI gadgets, including drawing occasions. It is likewise the string in which your application communicates with parts from the Android UI toolbox. Thusly, the principle string is likewise here and there called the UI string.

The framework does not make a different string for each case of a segment. All segments that keep running in a similar procedure are instantiated in the UI string, and framework calls to every part are dispatched from that string. Thus, techniques that react to framework callbacks, (for example, onKeyDown() to report client activities or a lifecycle callback strategy) dependably keep running in the UI string of the procedure.

For example, when the client touches a catch on the screen, your application's UI string dispatches the touch occasion to the gadget, which thusly sets its squeezed state and presents a nullify ask for on the occasion line. The UI string dequeues the demand and tells the gadget that it ought to redraw itself.

At the point when your application performs serious work in light of client cooperation, this single string model can yield poor execution unless you actualize your application appropriately. In particular, if everything is going on in the UI string, performing long tasks, for example, organize access or database questions will hinder the entire UI. At the point when the string is obstructed, no occasions can be dispatched, including drawing occasions. From the client's point of view, the application seems to hang. Far and away more terrible, if the UI string is obstructed for in excess of a couple of moments (around 5 seconds at present) the client is given the notorious "application not reacting" (ANR) exchange. The client may then choose to stop your application and uninstall it on the off chance that they are troubled.

Moreover, the Android UI toolbox isn't string safe. In this way, you should not control your UI from a specialist string—you should do all control to your UI from the UI string. Consequently, there are just two guidelines to Android's single string model:

1. Try not to hinder the UI string.
2. Try not to get to the Android UI toolbox from outside the UI string.

2.1.7 Multi-Tasking

An application for the most part contains numerous exercises. Every movement ought to be planned around a particular sort of activity the client can perform and can begin different exercises. For instance, an email application may have one movement to demonstrate a rundown of new email. At the point when the client chooses an email, another action opens to see that email. An action can even begin exercises that exist in different applications on the gadget. For instance, if the application needs to send an email, we can characterize aim to play out a "send" activity and incorporate a few information, for example, an email address and a message. An action from another application that announces itself to deal with this sort of purpose at that point opens. For this situation, the expectation is to send an email, so an email application's "create" action begins (if various exercises bolster a similar goal, at that point the framework gives the client a chance to choose which one to utilize). At the point when the email is sent, your movement resumes and it appears as though the email action was a piece of your application. Despite the fact that the exercises might be from various applications,

Android keeps up this consistent client encounter by keeping the two exercises in a similar errand. An assignment is a gathering of exercises that clients cooperate with when playing out a specific occupation. The exercises are organized in a stack (the "back stack"), in the request in which every action is opened. The gadget Home screen is the beginning spot for generally assignments. At the point when the client touches a symbol in the application launcher (or an easy route on the Home screen), that application's assignment goes to the frontal area. In the event that no assignment exists for the application (the application has not been utilized as of late), at that point another undertaking is made and the "fundamental" movement for that application opens as the root action in the stack.When the current activity starts another, the new action is pushed on the highest point of the stack and takes center. The past movement stays in the stack, yet is halted. At the point when an action stops, the framework holds the present condition of its UI. At the point when the client presses the BACK key, the present movement is flown from the highest point of the stack (the action is obliterated) and the past action continues (the past condition of its UI is reestablished). Exercises in the stack are never modified, just pushed and flew from the stack—pushed onto the stack when begun by the present movement and flew off when the client abandons it utilizing the BACK key. In that capacity, the back stack works as a "toward the end in, first out" protest structure. Figure 1 pictures this conduct with a course of events demonstrating the advance between exercises alongside the current back stack at each point in time. In the event that the client keeps on squeezing BACK, at that point every action in the stack is flown off to uncover the past one, until the point when the client comes back to the Home screen (or to whichever action was running when the undertaking started). At the point when all exercises are expelled from the stack, the undertaking never again exists.

An undertaking is a firm unit that can move to the "foundation" when clients start another errand or go to the Home screen, through the HOME key. While out of sight, every one of the exercises in the assignment are halted, yet the back stack for the errand stays in place—the undertaking has just lost concentration while another errand happens. An undertaking would then be able to come back to the "closer view" so clients can get the latest relevant point of interest. Assume, for instance, that the present errand (Task A) has three exercises in its stack—two under the present movement. The client presses the

HOME key, at that point begins another application from the application launcher. At the point when the Home screen shows up, Task A goes out of spotlight. At the point when the new application begins, the framework begins an undertaking for that application (Task B) with its own pile of exercises. In the wake of communicating with that application, the client returns Home again and chooses the application that initially began Task A. Presently, Task A goes to the closer view—every one of the three exercises in its stack are in place and the action at the highest point of the stack resumes.

Chapter -3 SYSTEM DEVELOPMENT

3.1 System Requirements

The structure and programming necessities for making Android applications using the Android SDK are according to the accompanying:

Windows XP (32-bit), Vista (32-or 64-bit), or Windows 7 (32-or 64-bit) Mac OS X 10.5.8 or later (x86 in a manner of speaking)

- Linux (attempted on Ubuntu Linux)
- GNU C Library (glibc) 2.7 or later is required.
- On Ubuntu Linux, adjustment 8.04 or later is required.
- 64-bit scatterings must be prepared for running 32-bit applications.

3.1.2 Supported Development Environments

- Eclipse IDE
- Eclipse 3.4 (Ganymede) or more important
- Eclipse JDT module (fused into most Eclipse IDE groups)
- Android-studio-bundle 162.4069837

For making Android applications, we recommend that you present one of these groups:

- Android-studio-bundle 162.4069837

Eclipse IDE for Java Developers

- Eclipse Classic (adjustments 3.5.1 and higher)
- Eclipse IDE for Java EE Developers

- JDK 5 or JDK 6 (JRE alone isn't sufficient) Android Development Tools module (required)

Other progression conditions or IDEs

- JDK 5 or JDK 6 (JRE alone isn't sufficient) Apache Ant 1.8 or later

3.1.3 Hardware requirement

The Android SDK requires disk storage for all of the components that you choose to install. The table below provides a rough idea of the disk-space requirements to expect, based on the components that you plan to use.

Component type	Approximate size	Comments
SDK Tools	35 MB	Required.
SDK Platform-tools	6 MB	Required.
Android platform (each)	150 MB	At least one platform is required.
SDK Add-on (each)	100 MB	Optional.
USB Driver for Windows	10 MB	Optional. For Windows only.
Samples (per platform)	10M	Optional.
Offline Documentation	250 MB	Optional.

Table 1 Components required by SDK

3.2 Material Design

You may have seen another look to your commonplace Google Apps. Drive, Gmail, Calendar and the Docs editors on Android gadgets are altogether redone to take after material outline standards, with extra updates crosswise over stages anticipated the coming months. This guide clarifies how these progressions will improve your experience utilizing these applications, and in the long run all Google items.

3.2.1 What is material design?

Google created material plan as another visual dialect for its items that is predictable crosswise over gadgets, so that as you change starting with one screen then onto the next, the experience feels the same. The emphasis is on making a steady and unsurprising background that alludes to our material world. For instance, we are for the most part comfortable with the way paper and ink carry on in reality. Parts of that material conduct are reflected in this new outline standard. nterface components act typically in this new visual dialect. For instance, tap a menu thing (e.g. All) to show the menu as a transitory sheet of paper that dependably covers the application bar and after that vanishes, rather than acting like an augmentation of the application bar.When you utilize an application, your activities have a beginning stage. Development from that point is smooth and unsurprising. When you touch the menu symbol to open the menu, it takes after your touch to open in a smooth, common way. When you touch screen components, they react with a dark touch expansive influence, giving an unmistakable reaction to your touch. Instead of force a similar format for each view, the new outline exploits every gadget design and adjusts as needs be. One of the key highlights of this new plan standard is consistency crosswise over stages. The client encounter is essentially the same on all gadgets and working frameworks. We are going for a bound together and solid experience over all Google things in the coming months. A material comparability is the coupling together hypothesis of a legitimized space and a strategy of advancement. The material is grounded in material reality, vivified by the examination of paper and ink, yet innovatively progressed and open to creative imperativeness and appeal. Surfaces and

edges of the material give visual signs that are grounded if all else fails. The utilization of consistent material characteristics helps clients rapidly comprehend affordances. However the adaptability of the material makes new affordances that supersede those in the physical world, without exasperating the gauges of physical science. The essentials of light, surface, and development are critical to passing on how questions move, connect, and exist in space and in connection to each other. Practical lighting demonstrates creases, separates space, and shows moving parts. The foundational components of print-based plan – typography, networks, space, scale, shading, and utilization of symbolism – manage visual medicines. These components do significantly more than satisfy the eye. They make pecking order, which means, and core interest. Ponder shading decisions, edge-to-edge symbolism, substantial scale typography, and deliberate void area make a strong and realistic interface that inundate the client in the experience. An accentuation on client activities makes center usefulness instantly clear and gives waypoints to the client. Movement regards and fortifies the client as the prime mover. Essential client activities are enunciation focuses that start movement, changing the entire plan. All move makes put in a solitary domain. Items are introduced to the client without breaking the congruity of experience even as they change and revamp.

.

Movement is important and suitable, serving to center consideration and look after progression. Criticism is unobtrusive yet clear. Advances are productive yet intelligent. The material condition is a 3D space, which implies all items have x, y, and z measurements. The z-pivot is oppositely adjusted to the plane of the show, with the positive z-hub reaching out towards the watcher. Each sheet of material involves a solitary position along the z-pivot and has a standard 1dp thickness, proportionate to one pixel of thickness on screens with a pixel thickness of 160.

On the web, the z-pivot is utilized for layering and not for viewpoint. The 3D world is copied by controlling the y-hub. Inside the material condition, virtual lights enlighten the scene. Key lights make directional shadows, while surrounding light makes delicate shadows from all points.

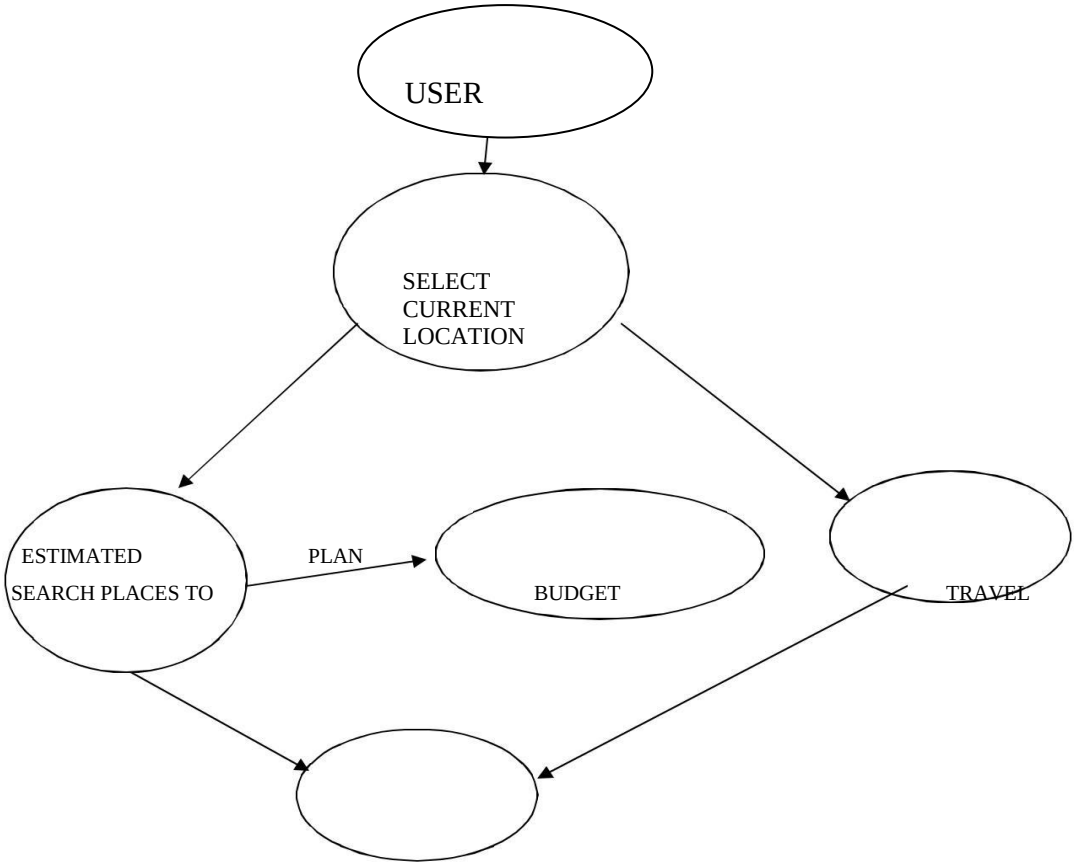
Shadows in the material condition are thrown by these two light sources. In Android improvement, shadows happen when light sources are hindered by sheets of material at different positions along the z-pivot. On the web, shadows are delineated by controlling the y-hub as it were. The accompanying case demonstrates the card with a stature of 6dp

3.2.2 Usability

A well designed product is available to clients of all capacities, incorporating those with low vision, visual deficiency, hearing debilitations, intellectual weaknesses, or engine impedances. Enhancing your item's availability improves the ease of use for everybody who utilizes it. It's likewise the correct activity. Material outline's worked in availability contemplations will enable you to oblige the majority of your clients. This area principally applies to versatile UI outline. For more data on planning and growing completely open items, visit the Google openness site.

3.3 Working of the app

1. USE CASE DIAGRAM



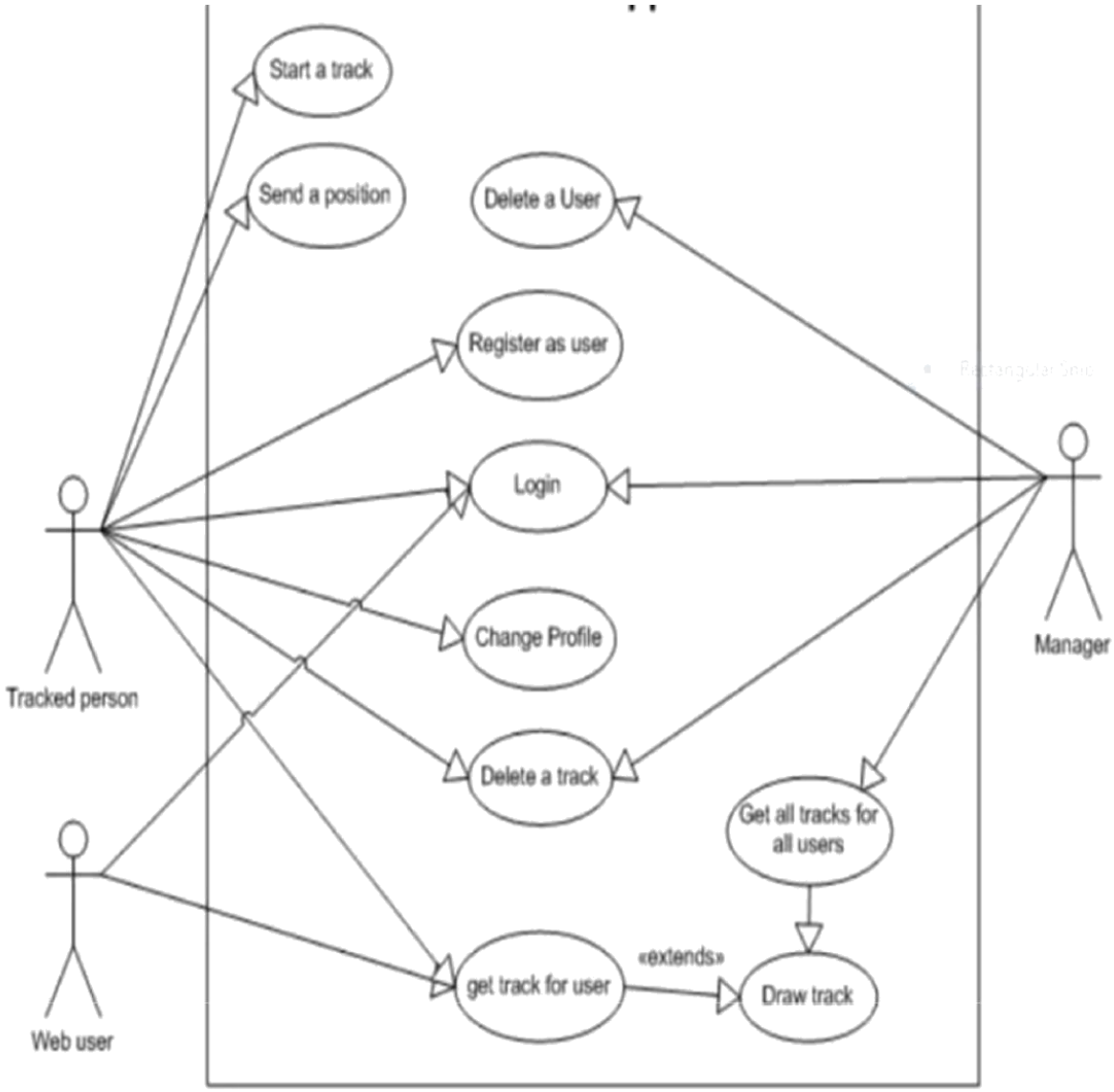


Figure 4 App interaction with user

2. Activity Diagram:(Login or New Registration)

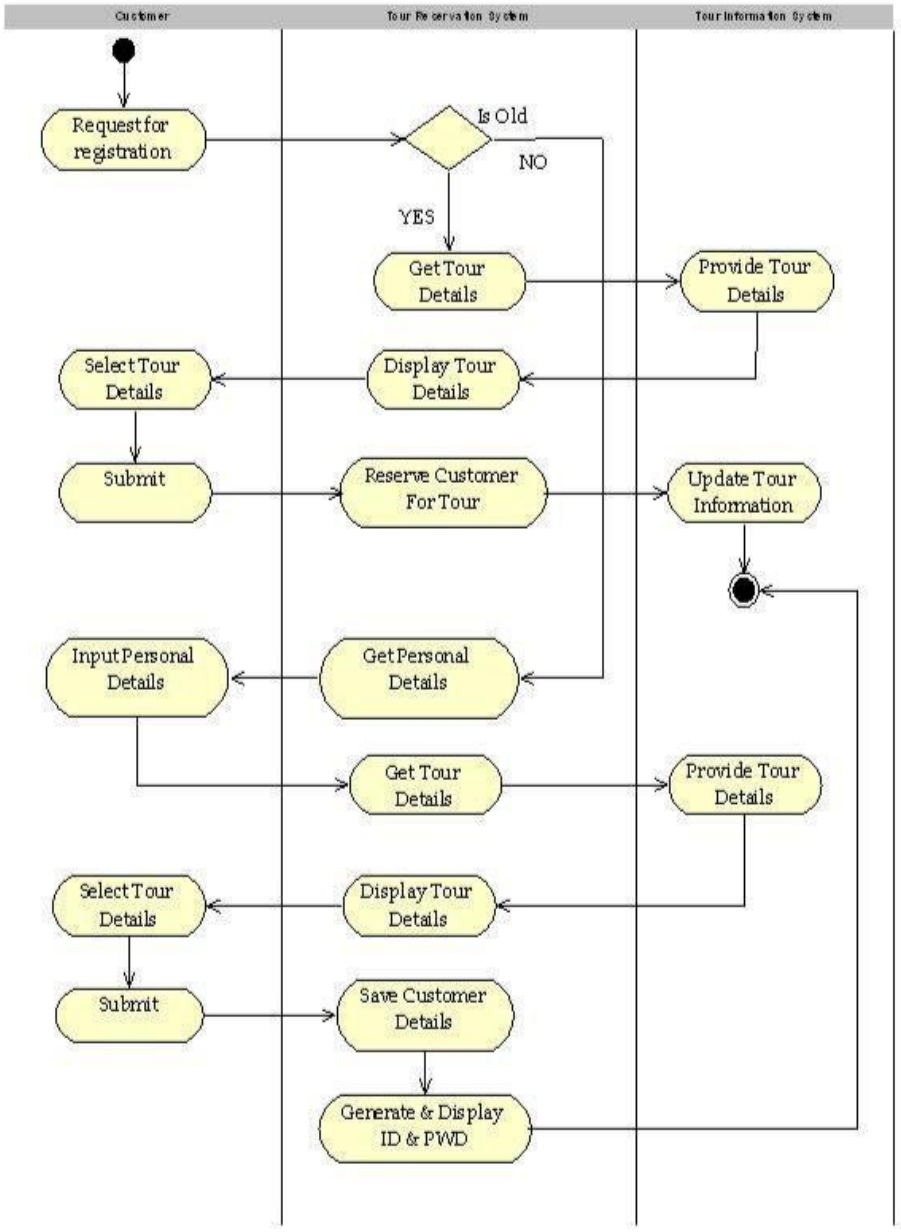


Figure 5 Graphical Representation of workflows

3. Collaboration Diagram:

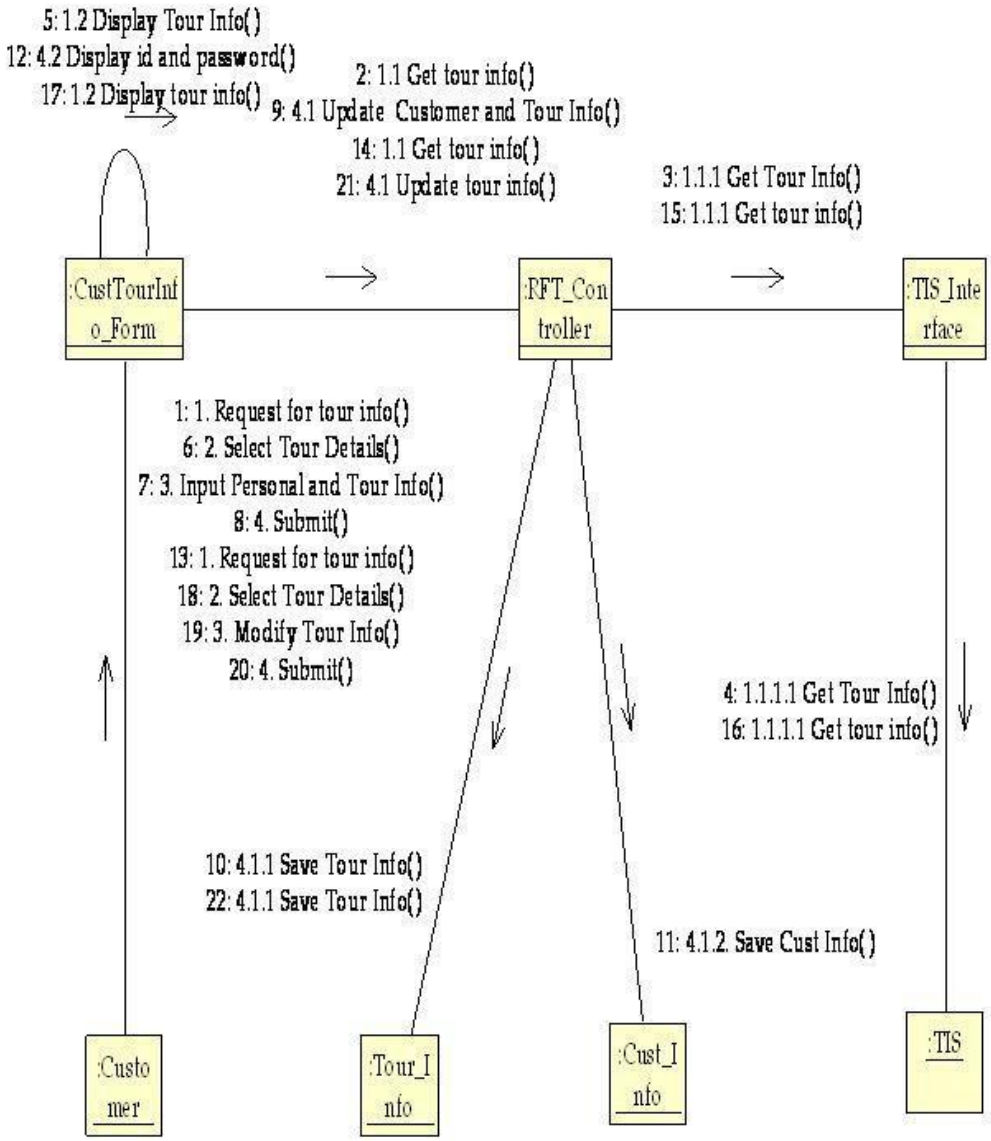


Figure 6 Interaction of components with each other

4. Component Diagram:

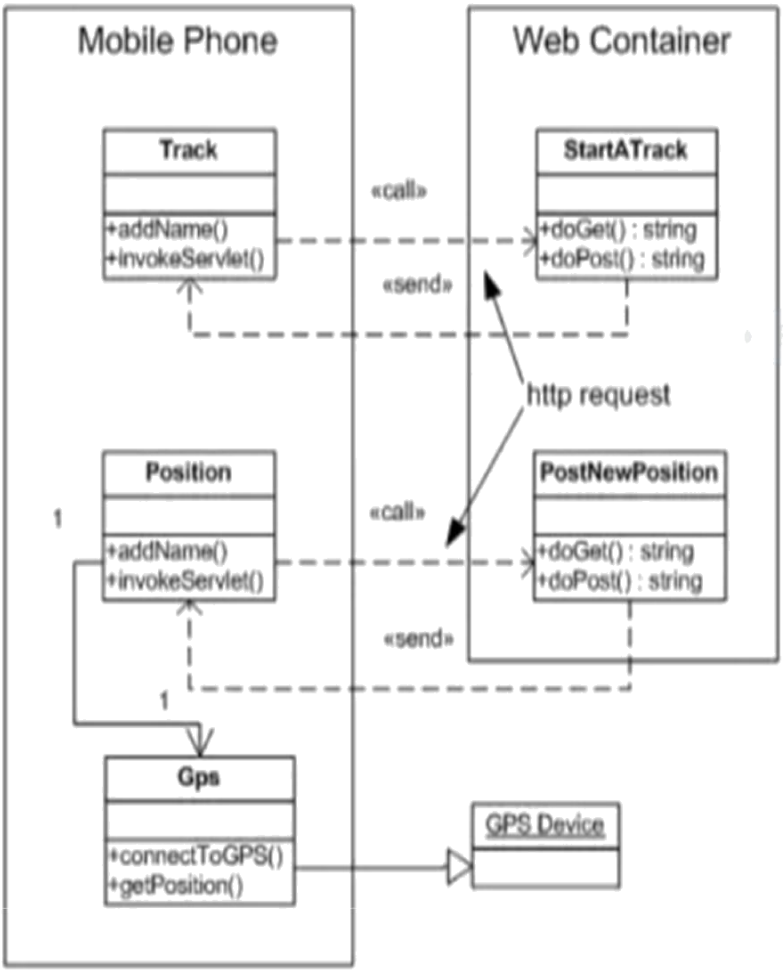


Figure 7 Physical aspects of system

5. Deployment Diagram:

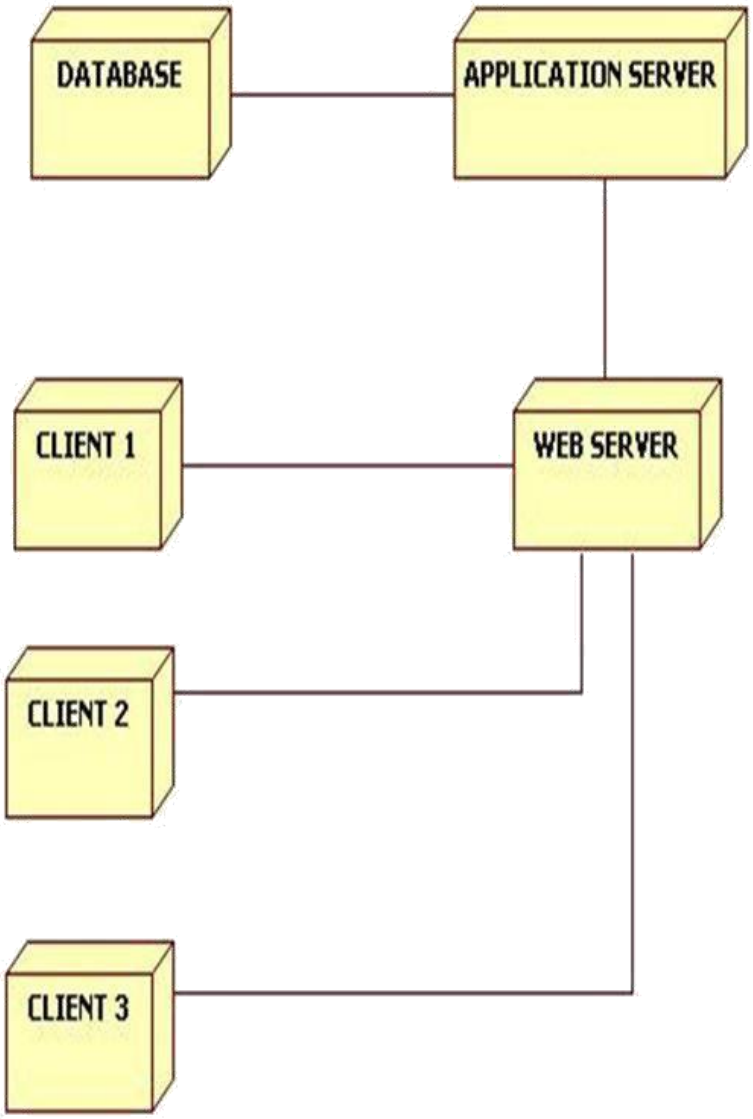


Figure 8 Hardware & Software Components existing

6. Sequence Diagram:

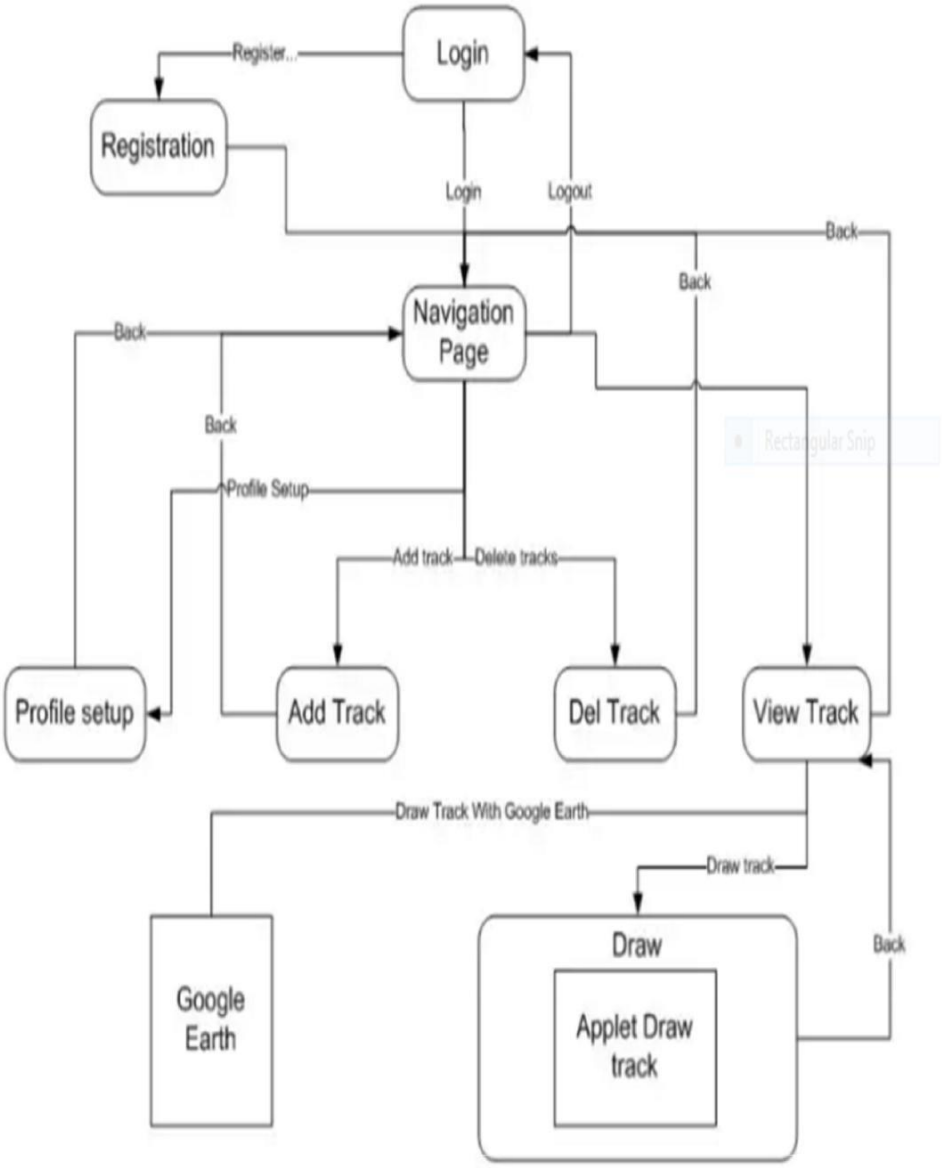


Figure 9

7. Class Diagram:

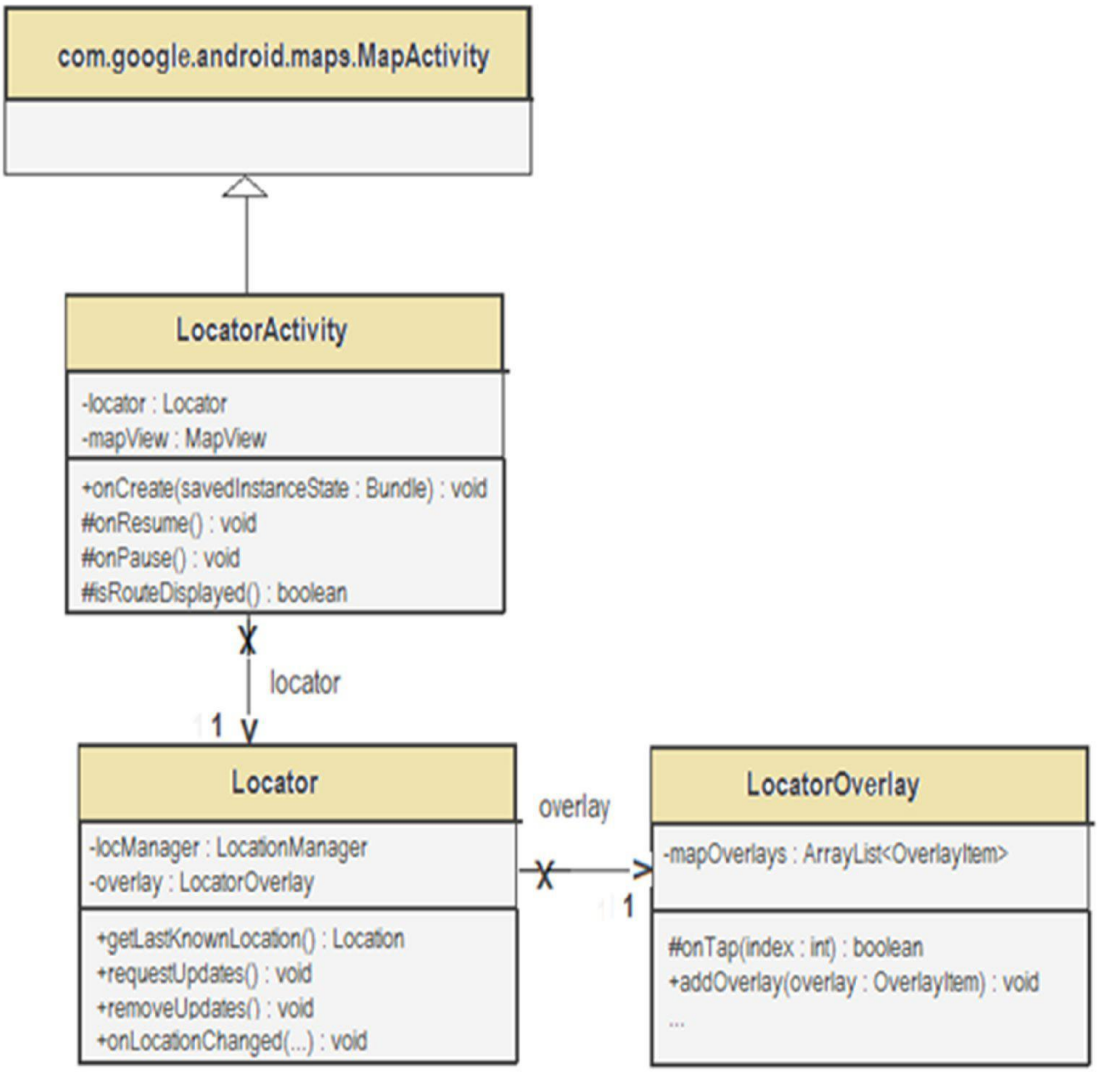


Figure 10 Different Aspects of system

8. Flow Diagram:

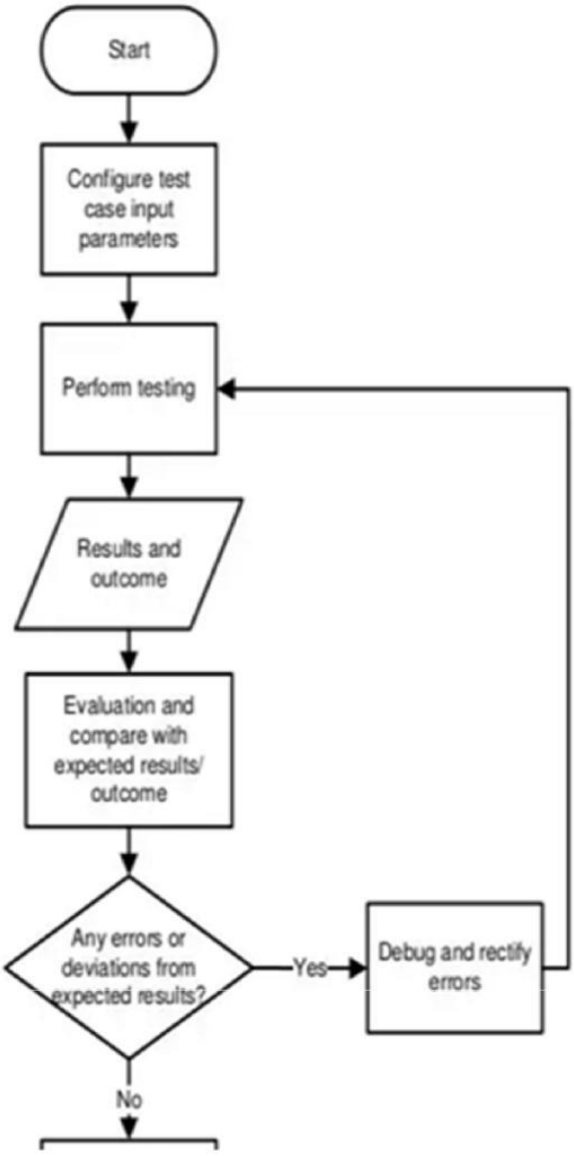


Figure 11 Reveals Underlined Structure

3.4 Screenshots:

1 Splash Screen: splash screen displayed on starting app. Uses animation to move the icons on the splash screen. The icons rotates at their own position.

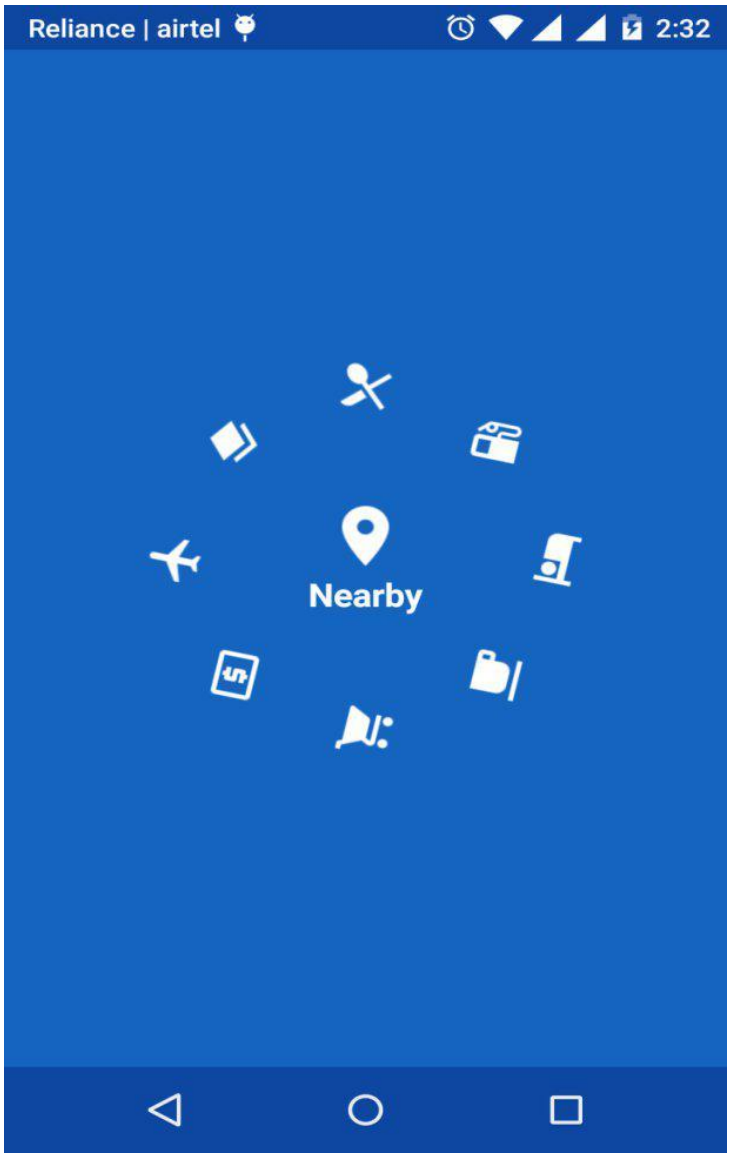


Figure 12

2. Categories: category list as tiles view, there are over 50 categories picked up from Google developers website from food to health which makes it easy for the user to locate any type of place of interest.

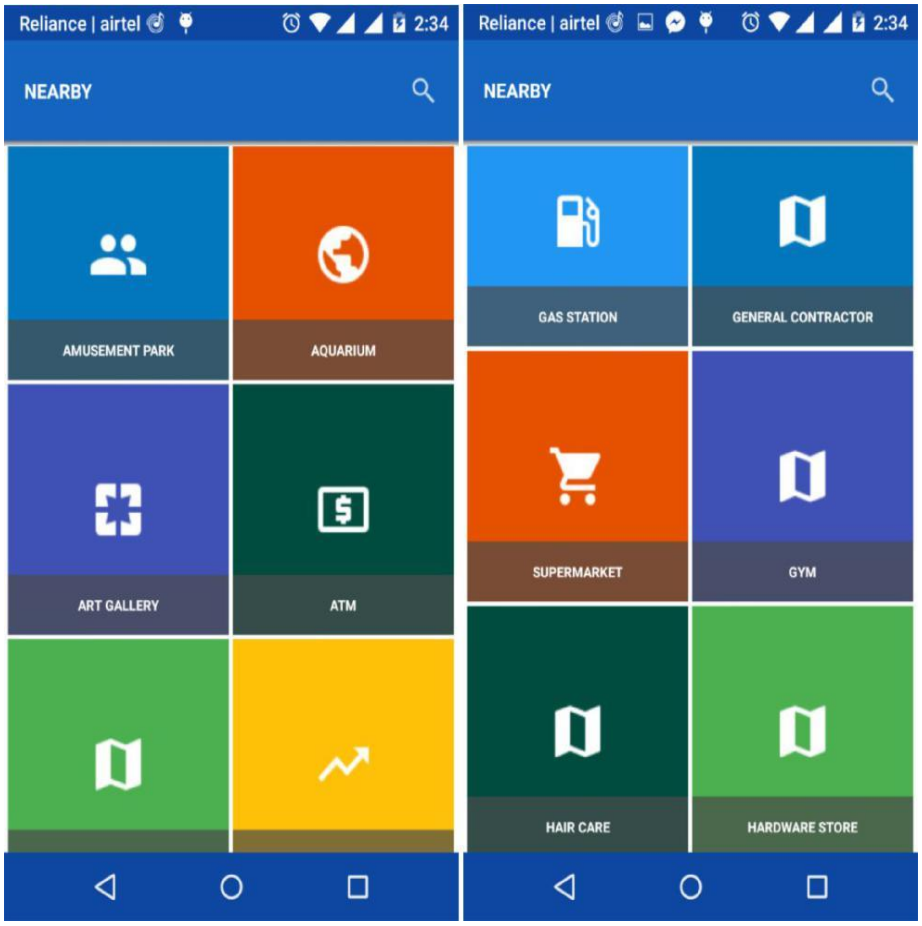


Figure 13

3. Details: on clicking cafe it list down the all the cafes within 10 km of range with its current status, rating and address. Uses material design concept to beautifully display the details. On selecting cafe it takes you to new activity which displays its position on map its reviews and gallery. It also displays two icons on map one for directions and other for Google maps page

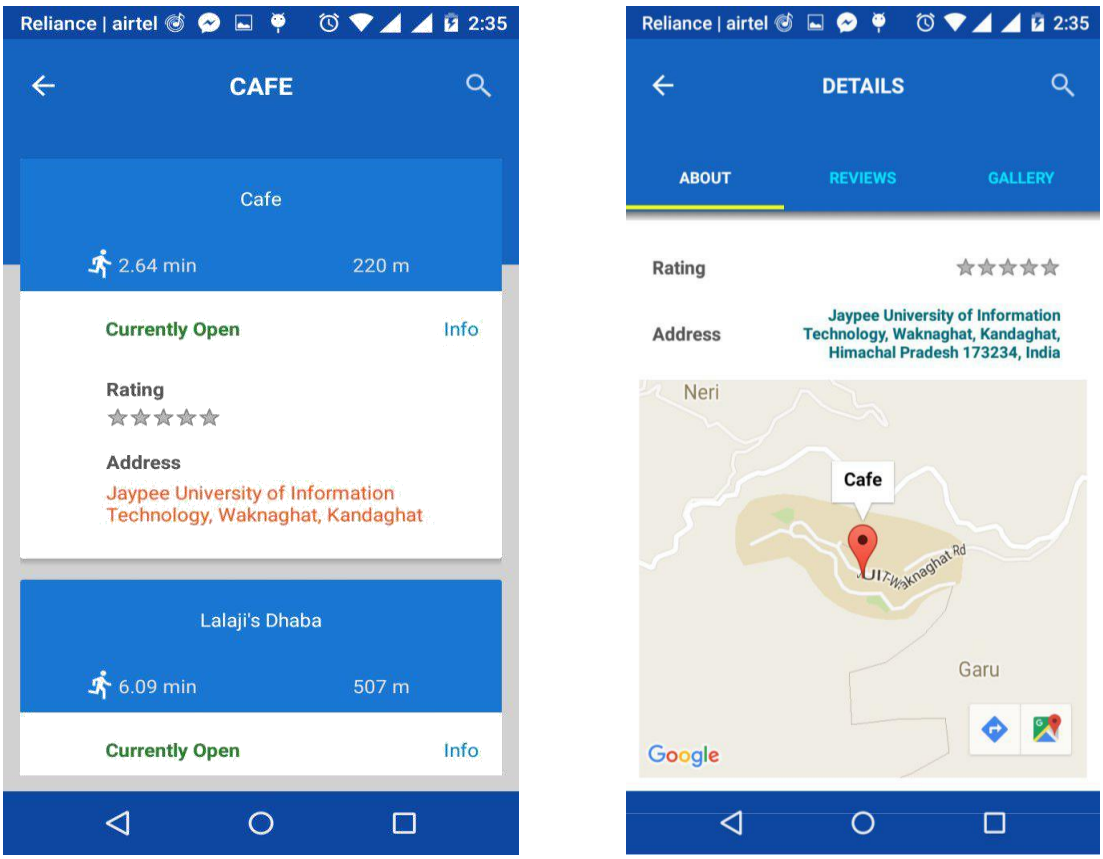


Figure 14

4. Maps: Shows directions for the place selected. It shows paths in yellow color and also distance and time by foot, car or train if available.

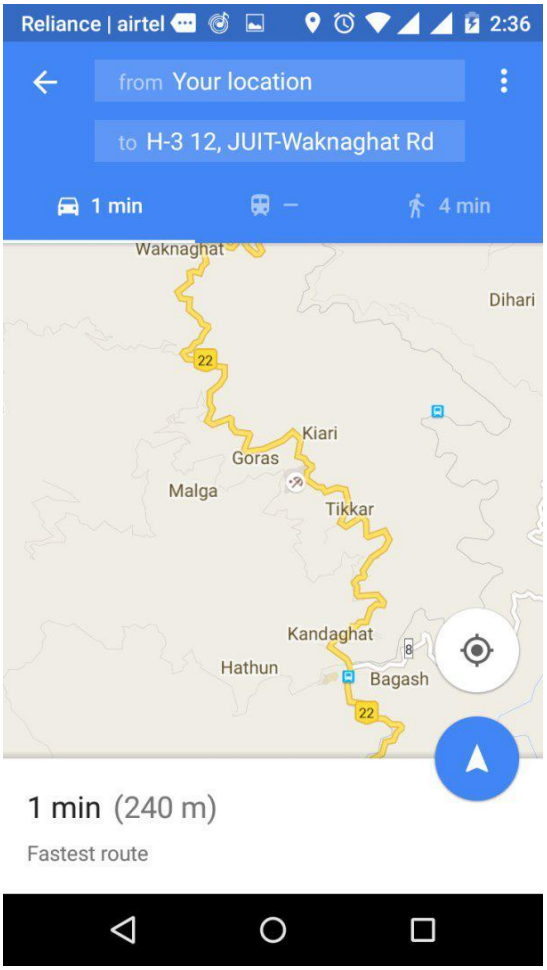


Figure 15

5. Searching: search option with auto complete facility and on clicking yellow sign it shows direction on Google maps. User can search any place whether in locality or not, it can be any place in any part of the earth.

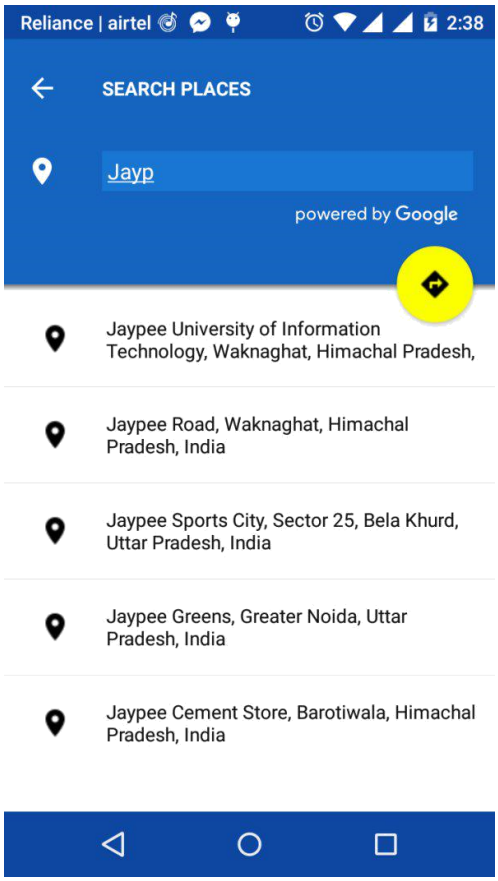


Figure 16

6. Gallery: gallery activity shows images of selected place. It picks up pictures from Google database if anyone has uploaded by tagging that place.

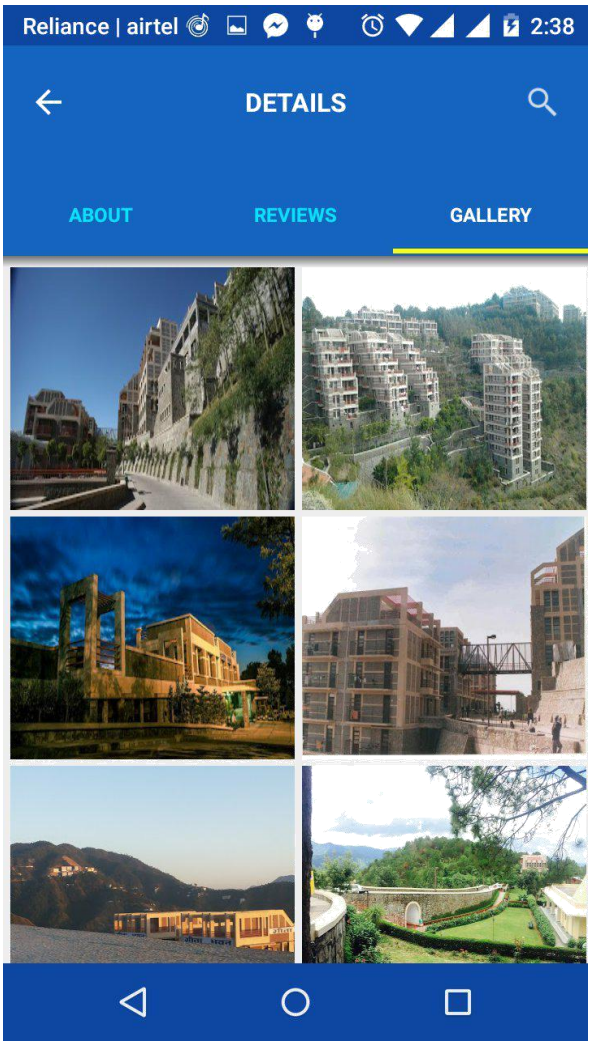


Figure 17

Chapter-4 PERFORMANCE ANALYSIS

4.1 Testing Fundamentals

Android gives an organized structure that energizes you test all parts of your application. The Android stage and Testing Support Library consolidate gadgets and APIs for setting up and running test applications inside an emulator or physical contraption.

This file guides you through key thoughts related to Android application testing, and gives an outline of the testing gadgets and APIs made by Google. If you have to keep away from the sensible chart, and start making sense of how to make and run your tests using these APIs and gadgets, go to Getting started with testing in Android Studio. In case you are not using Android Studio, go to Testing from the summon line

.

Testing Concepts

Android testing depends on JUnit. When all is said in done, a JUnit test is a technique whose announcements test a piece of the application. You sort out test techniques into classes called test cases, and gathering experiments into test suites.

In JUnit, you manufacture at least one test classes and utilize a test sprinter to execute them on your nearby machine. With Android Studio, you can assemble at least one test source records into an Android test application and utilize it to test your application on the Emulator or physical Android gadget. The structure of your test code and the way you manufacture and run the tests in Android Studio rely upon the sort of testing you are performing. The accompanying table outlines the basic testing composes for Android:

Type	Subtype	Description
Unit tests	Local Unit Tests	Unit tests that run on your local machine only. These tests are compiled to run locally on the Java Virtual Machine (JVM) to minimize execution time. Use this approach to run unit tests that have no dependencies on the Android framework or have dependencies that mock objects can satisfy.
	Instrumented unit tests	Unit tests that run on an Android device or emulator. These tests have access to `Instrumentation` information, such as the `Context` of the app you are testing. Use this approach to run unit tests that have Android dependencies which mock objects cannot easily satisfy.
Integration Tests	Components within your app only	This type of test verifies that the target app behaves as expected when a user performs a specific action or enters a specific input in its activities. For example, it allows you to check that the target app returns the correct UI output in response to user interactions in the app's activities. UI testing frameworks like Espresso allow you to programmatically simulate user actions and test complex intra-app user interactions.
	Cross-app Components	This type of test verifies the correct behavior of interactions between different user apps or between user apps and system apps. For example, you might want to test that your app behaves correctly when the user performs an action in the Android Settings menu. UI testing frameworks that support cross-app interactions, such as UI Automator, allow you to create tests for such scenarios.

Table 1 Testing types

Instrumentation

Android instrumentation is an arrangement of control strategies, or snares, in the Android framework. These snares control an Android part freely of its ordinary lifecycle. They likewise control how Android loads applications.

The accompanying outline abridges the testing system

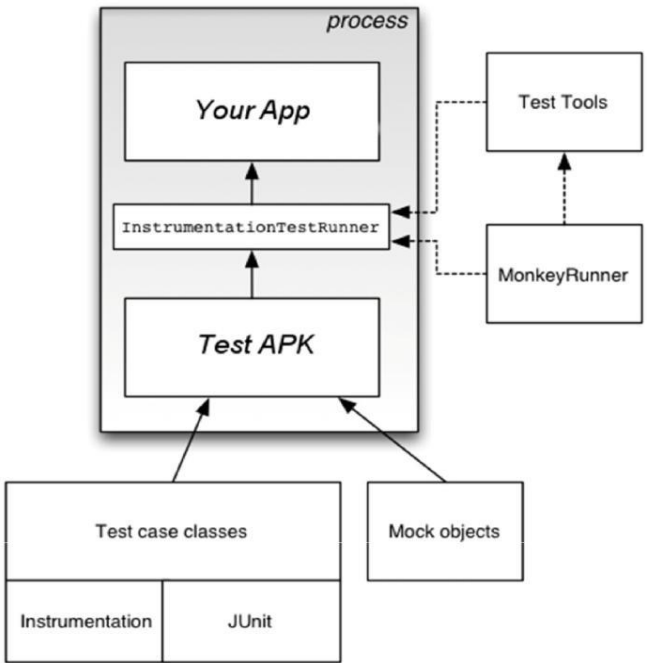


Figure 18

Regularly, an Android part keeps running in a lifecycle that the framework decides. For instance, an Activity protest's lifecycle begins when an Intent actuates the Acticity. The framework calls the protest's oncreate() technique, on then the onResume() strategy. At the point when the client begins another application, the framework calls the onPause() strategy. On the off chance that the Activity code calls the complete() strategy, the framework calls the onDestroy() technique. The Android system API does not give a path to your code to conjure these callback strategies specifically, however you can do as such utilizing instrumentation.

The framework runs every one of the segments of an application in a similar procedure. You can permit a few parts, for example, content suppliers, to keep running in a different

procedure, however you ordinarily can't compel an application onto an indistinguishable procedure from another running application.

Instrumentation tests, in any case, can stack both a test APK of your test classes and your application's APK into a similar procedure. Since the segments of your application and their tests are in a similar procedure, your tests can summon techniques, and alter and inspect fields in your application.

Testing Source Sets:

When you create a new app module, Android Studio creates the src/test/ and src/androidTest/ source set directories for you. Place the test classes you want to run locally on your machine in the test/ source set and the test classes you want to run on an actual Android device in the androidTest/ source set. Gradle uses the androidTest/source set when generating the test APK you use to test your app. To learn more about build variants and source sets, read the Configure your build overview.

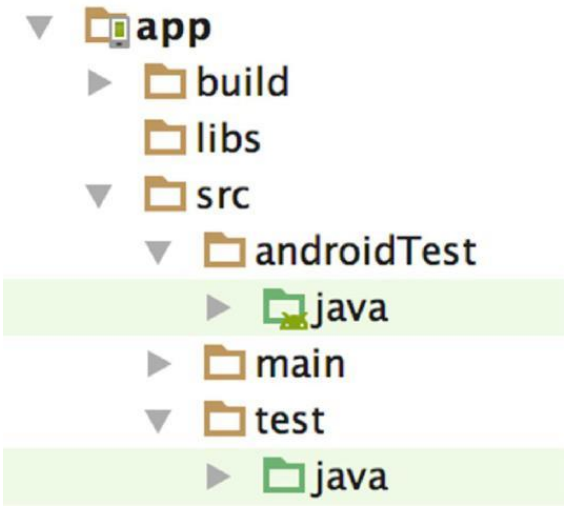


Figure 19 Default App Module Test Directory

Testing APIs:

The accompanying rundown condenses the normal APIs identified with application testing for Android.

JUnit

You should write your unit or integration test class as a JUnit 4 test class. JUnit is the most popular and widely-used unit testing framework for Java. The framework offers a convenient way to perform common setup, teardown, and assertion operations in your test.

JUnit 4 allows you to write tests in a cleaner and more flexible way than its predecessor versions. Unlike the previous approach to Android unit testing based on JUnit 3, with JUnit 4, you do not need to extend the junit.framework.TestCase class. You also do not need to propend the test keyword to your test method name, or use any classes in thejunit.framework or junit.extensions package.

A basic JUnit 4 test class is a Java class that contains one or more test methods. A test method begins with the @Test annotation and contains the code to exercise and verify a single functionality (that is, a logical unit) in the component that you want to test.

The following snippet shows an example JUnit 4 integration test that uses the Espresso APIs to perform a click action on a UI element, then checks to see if an expected string is displayed.

```
@RunWith(AndroidJUnit4.class)
@LargeTest
public class MainActivityInstrumentationTest {
@Rule

public ActivityTestRule mActivityRule = new ActivityTestRule<>( MainActivity.class);
@Test
public void sayHello(){
onView(withText("Say hello!")).perform (click());
onView(withId(R.id.textView)).check(matches(withText("Hello, World!")));
}}
```

In your JUnit 4 test class, you can call out areas in your test code for unique handling by utilizing the accompanying comments:

@Before: Use this explanation to determine a square of code that contains test setup activities. The test class summons this code hinder before each test. You can have multiple@Before techniques however the request in which the test class calls these strategies isn't ensured.

@After: This explanation determines a piece of code that contains test tear-down tasks. The test class calls this code hinder after each test strategy. You can characterize numerous @After activities in your test code. Utilize this explanation to discharge any assets from memory.

@Test: Use this comment to stamp a test technique. A solitary test class can contain numerous test techniques, each prefixed with this comment.

@Rule: Rules enable you to adaptably include or reclassify the conduct of each test strategy reusably. In Android testing, utilize this comment together with one of the test decide classes that the Android Testing Support Library gives, for example, Activity Test Rule or ServiceTestRule.

@BeforeClass: Use this explanation to determine static techniques for each test class to summon just once. This testing step is helpful for costly tasks, for example, interfacing with a database.

@AfterClass: Use this explanation to determine static techniques for the test class to conjure simply after all tests in the class have run. This testing step is helpful for discharging any assets distributed in the @BeforeClass square.

@Test(timeout=<milliseconds>): Some comments bolster the capacity to go in components for which you can set qualities. For instance, you can determine a timeout period for the test. In the event that the test begins however does not finish inside the

given timeout period, it naturally falls flat. You should determine the timeout period in milliseconds, for example:@Test(timeout=5000).

For more comments, see the documentation for JUnit Annotations and the Android-Specific Annotations.

You utilize the JUnit Assert class to confirm the accuracy of a question's state. The affirm strategies look at values you anticipate from a test to the genuine outcomes and toss a special case if the examination comes up short. Affirmation Classes depicts these strategies in more detail.

Android Testing Support Library APIs

The Android Testing Support Library gives an arrangement of APIs that enable you to rapidly manufacture and run test code for your applications, including JUnit 4 and useful UI tests. The library incorporates the accompanying instrumentation-based APIs that are valuable when you need to computerize your tests:

- AndroidJUnitRunner
- A JUnit 4-good test sprinter for Android.
- Espresso
- A UI testing structure; reasonable for useful UI testing inside an application.
- UI Automator
- A UI testing structure reasonable for cross-application useful UI testing between both framework and introduced applications.

Assertion classes

Since Android Testing Support Library APIs expand JUnit, you can utilize statement strategies to show the consequences of tests. An attestation strategy thinks about a real esteem returned by a test to a normal esteem, and tosses an Assertion Exception if the examination test fizzles. Utilizing affirmations is more advantageous than logging, and gives better test execution.

To streamline test advancement, you should utilize the Hamcrest library, which gives you a chance to make more adaptable tests utilizing the Hamcrest matcher APIs. Monkey And MonkeyRunner

The SDK gives two instruments to useful level application testing:

Monkey

This is a summon line mechanical assembly that sends pseudo-self-assertive surges of keystrokes, touches, and movements to a device. You run it with the Android Debug Bridge (adb) mechanical assembly, and use it to weight test your application, report back goofs any that are experienced, or go over a surge of events by running the gadget various conditions with a comparable sporadic number seed. monkeyrunner

This gadget is an API and execution condition for test programs written in Python. The API joins capacities with respect to partner with a contraption, presenting and uninstalling packs, taking screen catches, differentiating two pictures, and running a test package against an application.

Using the API, you can make a broad assortment out of considerable, powerful, and complex tests. You run programs that usage the API with the monkeyrunner arrange line gadget.

Chapter-5 CONCLUSION

5.1) CONCLUSION

Location based Services offer various inclinations to the compact customers to recuperate the information about their present zone and process that data to get more significant information near their zone. With the help of A-GPS in phones and through Web Services using GPRS, Location build Services can be executed in light of Android based propelled cell phones to give these regard included organizations: training clients in regards to current movement conditions, giving guiding information, helping them discover near to lodgings.

Location Based Services is an Android App , enabling travelers to plan and book the perfect trip. Location Based Services offers users an easy and simple interface helping them to find places. Upon using the app the user can simply select the city he wants to visit and nearby places.

On selecting the city the user will come across a list of places. These places are the most famous places in that area. They can be historical monuments like forts, palaces etc.. The user can select any place of his personal liking and then can gather information about that place with the help of a short article or with the pictorial information provided at the interface.

The main purpose of this app is to make the travel experience of the user easy and simple by providing reliable and easy information to the user about the places near him.

In this project we have tried to implement all the above functions with the help of the android app.

5.2) FUTURE SCOPE

For future we can have the following features in the app which will be user friendly and will help the user to easily go to place he wants to visit.

- Add more cities
- Add more place in each city
- Ask for specific travel details
- Include Review for Hotels
- Include reviews for mode of transport
- Ask for user preferences on which more cities to add
- Ask for user preferences places to be added in the existing cities
- Log in feature for users
- Discount for regular customers

We want to build an app that simplifies the entire user experience of a person who is visiting a new city by providing him complete details of the entire list of places he wants to visit and provide the user with a happy and memorable trip.

In the end we would like to publish our app on the play store.

5.2.1) TECHNOLOGIES TO BE IMPLEMENTED IN FUTURE

1. WEB CRWALING

A Web crawler is an Internet bot which systematically browses the World Wide Web, typically for the purpose of Web indexing. A Web crawler may also be called a Web spider, an ant, an automatic indexer, or (in the FOAF software context) a Web scutter.

Web look for mechanical assemblies and some remarkable objectives utilize Web creeping or spidering programming to resuscitate their web substance or reports of others regions' web content. Web crawlers can duplicate every single one of the pages they visit

for later preparing by a web look mechanical assembly which records the downloaded pages so the clients can search for altogether more proficiently.

Crawlers can avow hyperlinks and HTML code. They can correspondingly be utilized for web scratching

Overview

A Web crawler begins with a rundown of URLs to visit, called the seeds. As the crawler visits these URLs, it recognizes every one of the hyperlinks in the page and adds them to the rundown of URLs to visit, called the creep boondocks. URLs from the outskirts are recursively gone by as indicated by an arrangement of approaches. In the event that the crawler is performing documenting of sites it duplicates and spares the data as it goes. The chronicles are generally put away in such a way they can be seen, perused and explored as they were on the live web, however are saved as 'depictions'.

The expansive volume suggests the crawler can just download a predetermined number of the Web pages inside a given time, so it needs to organize its downloads. The high rate of progress can suggest the pages may have just been refreshed or even erased.

The quantity of conceivable URLs crept being produced by server-side programming has likewise made it troublesome for web crawlers to abstain from recovering copy content. Unlimited blends of HTTP GET (URL-based) parameters exist, of which just a little determination will really return one of a kind substance. For instance, a straightforward online photograph display may offer three choices to clients, as indicated through HTTP GET parameters in the URL. In the event that there exist four approaches to sort pictures, three decisions of thumbnail estimate, two record groups, and a choice to incapacitate client gave content, at that point a similar arrangement of substance can be gotten to with 48 unique URLs, which may all be connected on the site. This numerical blend makes an issue for crawlers, as they should deal with unlimited mixes of generally minor scripted changes keeping in mind the end goal to recover novel substance.

Crawling strategy

The conduct of a Web crawler is the result of a mix of strategies:

1) A choice strategy which expresses the pages to download,

2) A return to approach which states when to check for changes to the pages.3) A politeness policy that states how to avoid overloading Web sites, and

4) A parallelization policy that states how to coordinate distributed web crawlers.

References

Journal Article

[1] Manav Singhal and Anupam Shukla, "Implementation of Location based Services in Android using GPS and Web Services", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2, January 2012.

Thesis

[2] Isha Sahu and Ishita Chakraborty, "Understanding Location Manager in Android and Implementing an Optimal Image Geotagging Application", (IJCTT) – volume 4 Issue 6– Month 2013.

Journal Article

[3] Alexander Troshkov, Kirill Kulakov," Tourist Application for Mobile Platforms", Petrozavodsk State University Petrozavodsk, Russia.

Online

[4] Qusay H. Mahmoud,"J2ME and Location based Services" - March 2004 http://developers.sun.com/mobility/apis/articles/location.

[5] ValerieBennett,"LocationBasedServices",http://www.ibm.com/developerworks/ibm/library/i-lbs.

Book

[6] Shane Condor and Lauren Darcy ,"Android Wireless Application Development".

Websites

[7] Google Places API

http://code.google.com/apis/maps/documentation/places/

[8] Google places API

http://code.google.com/apis/maps/documentation/imageapis/index.html

[9] Google Places API