PROJECT REPORT

To create a 'Book My Show' type app for Grasim plant's movie theatre

Made by –

NAME	ID NUMBER	DISCIPLINE
Akshat Jain	2017A7PS0110P	CS
Trivendra Singh	2017B4TS1227P	Math
Pranjal Choubey	2017A7PS0034G	CS
Amrutha Dude	2017A7PS0259H	CS

Prepared in partial fulfilment of the Practice School – I
Course No. BITS F221
At
Grasim Stable Fiber Division, Nagda
At the PS-1 station of
Birla Institute of Technology and Science, Pilani
(July 2019)

ABSTRACT

BIRLA INSTITUTE OF TECHNOLOGYAND SCIENCE, PILANI (RAJASTHAN) PRACTICE SCHOOL DIVISION

STATION : Grasim Industries, Nagda

DURATION: May 21, 2019 – July 13, 2019

DATEOF SUBMISSION: July 8, 2019

TITLE OF REPORT: TO DEVLOP A 'BOOK MY SHOW ' TYPE ANDROID APP FOR MOVIE THEATRE

DETAILS OF FACULTY: **Mr. Rakesh Chauhan**

NAME OF PS FACULTY: **Mr. Kamlesh Tiwari**

KEY WORDS: Android App Development, Java , MySQL, Application Framework, Integrated Browser, Optimized Graphics, SQLite, Handset Layouts, Data Storage, Connectivity, Server Monitoring, Virtual Machine

PROJECT AREAS: Android Development, Database Management, Connectivity, Access Control, Password Policy, Master Data Management, Admin Control, Integrated Development Environment (IDE), Software Development

ABSTRACT

This report attempts to summarise the work done and concepts learnt as a PS-I intern at Department of Information Technology, Grasim Industries, Nagda. The 20 days practice school programme gave me an opportunity to see, experience and learn the role and contribution of IT department in a chemical plant. This report first discusses the backbone of IT in any organisation, IT infrastructure. IT infrastructure mainly deals with Active Directory which is responsible for access control, information security and asset management. This report gives a brief overview of Android Development and its implementation in Grasim Industries. It also briefs the structure and development of the android application and tells about the its various features.

Signature of PS Faculty:
(Mr. Kamlesh Tiwari}
Date: July 13, 2019.

FEEDBACK BY MENTOR

All the members of this group were regular and the work was divided equally. Their job was completed on time and was of satisfactory quality. They did not misuse the privileges given to them and kept me informed of their progress regularly.

I am satisfied with their work. It was good to mentor and guide them throughout their project work. I wish them best for their future endeavors.

Mr. Rakesh Chouhan

(IT department Grasim)

ACKNOWLEDGEMENT

I wish to express my gratitude towards Ms. Nikhita Naik, HR Coordinator for Practice School- I, BITS-Pilani for her enthusiastic support, cooperation and help. I wish to thank her for facilitating this programme which gave me an opportunity to learn and train in IT Department of GRASIM INDUSTRIES, NAGDA. I sincerely thank the staff of IT Department, especially Mr Rakesh Chouhan, for his time, support, guidance and encouragement.

I express my gratitude towards Mr. Kamlesh Tiwari our PS-I faculty for his guidance and support throughout the training here. I also wish to thank the Practice School Division of BIRLA INSTITUTE OF TECHNOLOGYAND SCIENCE, PILANI for giving us the opportunity to train and learn at GRASIM INDUSTRIES, NAGDA.

INDEX

INTRODUCTION

The Information Technology department at Grasim Industries mainly does the job of providing technical and functional support and facilitating sales and distribution activities. Its job is also to ensure that plants run smoothly without any glitches. The IT department implements purchased software and trains employees in its efficient and judicious use. The IT department coordinates data sharing between different departments and ensures data consistency. The IT department is responsible for the network connectivity in the plant premises. It also looks into asset management, specifically, providing access and authorization to various user accounts on all the computers or laptops in the Stable Fiber Division.

Android is a software platform and operating system for mobile devices based on the Linux operating system and developed by Google and the Open Handset Alliance. It allows developers to write managed code in a Java-like language that utilizes Google-developed Java libraries ,but does not support programs developed in native code. The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 34 hardware, software and telecom companies devoted to advancing open standards for mobile devices. When released in 2008, most of the Android platform will be made available under the Apache free-software and open-source license

The IT department in Grasim Industries plays a non-core operational role. It basically helps in the smooth and efficient running of production processes. I got to learn about the organisational infrastructure of a giant company and the role of IT department in setting it up and maintaining it. I was involved in the routine day-to-day activities of this department and learnt about Android Development. This report first introduces the Android Development Concepts and then explains the Application Framework and the complexities that follow it..

Android Development

1. THE BIRTHOFANDROID

1.1.1. Google Acquired Android Inc.

In July2005, Google acquired Android Inc., a small startup company based in Palo Alto, CA. Android's co-founders who went to work at Google included Andy Rubin (co-founder of Danger), Rich Miner (co-founder of Wildfire Communications, Inc), Nick Sears(once VPat T-Mobile), and Chris White (one of the first engineers at WebTV).At the time, little was known about the functions of Android Inc. other than they made software for mobile phones. At Google, the team, led by Rubin, developed a Linux - based mobile devices which they marketed to handset makers and carriers on the premise of providing flexible, upgradeable system. It was reported that Google had already 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.

2. Open Handset Alliance Founded

On5 November 2007,the Open Handset Alliance, a consortium of several companies which include Google, HTC ,Intel, Motorola, Qualcomm, T-Mobile ,Sprint Nextel and NVIDIA, was unveiled with the goal to develop open standards for mobile devices. Along with the formation of the Open Handset Alliance,the OH also unveiled their first product, Android, an open source mobile device platform based on the Linux operating system.

3. Hardware

Google has unveiled at least three prototypes for Android, at the Mobile World Congress on February 12, 2008. One prototype at the ARM booth displayed several basic Google applications. A 'd-pad' control zooming of items in the dock with a relatively quick response. A prototype at the Google IO conference on May 28, 2008 had a 528 MHz QUALCOMM processor and a Synaptic capacitive touch screen, and used the UMTS cellular standard. It had128 MB of RAM and 256 MB of flash, showing that Android's memory requirements are reasonable. The demo was carried out using a 3.6MbpsHSDPAconnection.

2. FEATURES

1. Application Framework

It is used to write applications for Android. Unlike other embedded mobile environments, Android applications are all equal, for instance, an applications which come with the phone are no different than those that any developer writes. The framework is supported by numerous open source libraries such as pens', SQ Lite and lib c. It is also supported by the Android core libraries. From the point of security, the framework is based on UNIX file system permissions that assure applications have only those abilities that mobile phone owner gave them at install time.

2. Dalvik Virtual Machine

It is extremely low-memory based virtual machine, which was designed especially for Android to run on embedded systems and work well in low power situations. It is also
tuned to the CPU attributes. The Dalvik VM creates a special file format (.DEX) that is created through build time post processing. Conversion between Java classes and. DEX format
is done by included “dx” tool.

3. Integrated Browser

Google made a right choice on choosing Web Kit as open source web browser. They added a two pass layout and frame flattening. Two pass layout loads a page without waiting for blocking elements, such as external CSS or external Java Script and after
a while renders again with all resources downloaded to the device. Frame flattening converts founded frames into single one and loads into the browser. These features increase speed and usability browsing the internet via mobile phone.

4. Optimized Graphics

As Android has 2D graphics library and 3D graphics based on Open GLES1.0 ,possibly we will see great applications like Google Earth and spectacular games like Second Life, which come on Linux version. At this moment, the shooting legendary 3D game Doom was presented using Android on the mobile phone

5. SQLite

Extremely small (< 500kb) relational data base management system, is integrated in Android. It is based on function calls and single file, where all definitions, tables and data are stored. This simple designs more than suitable for a platform such as Android.

6. Data Storage

SQ Lite is used for structured data storage. SQ Lite is a powerful and lightweight relational data base engine available to all applications.

2. DETAILEDDESCRIPTIONOFTHETOPIC

2.1 OPERATION

2.1.1. 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.

2.1.2. 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. It helps to manage security, memory management, process management, network stack and other important issues. Therefore, the user should bring Linux in his mobile device as the main operating system and install all the drivers required in order to run it. Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework).This same mechanism allows components to be replaced by the user. Underlying all applications is a set of services and systems.

2.2 ARCHITECTURE

The following diagram shows the major components of the Android operating system. Each section isdescribed in more detail below.

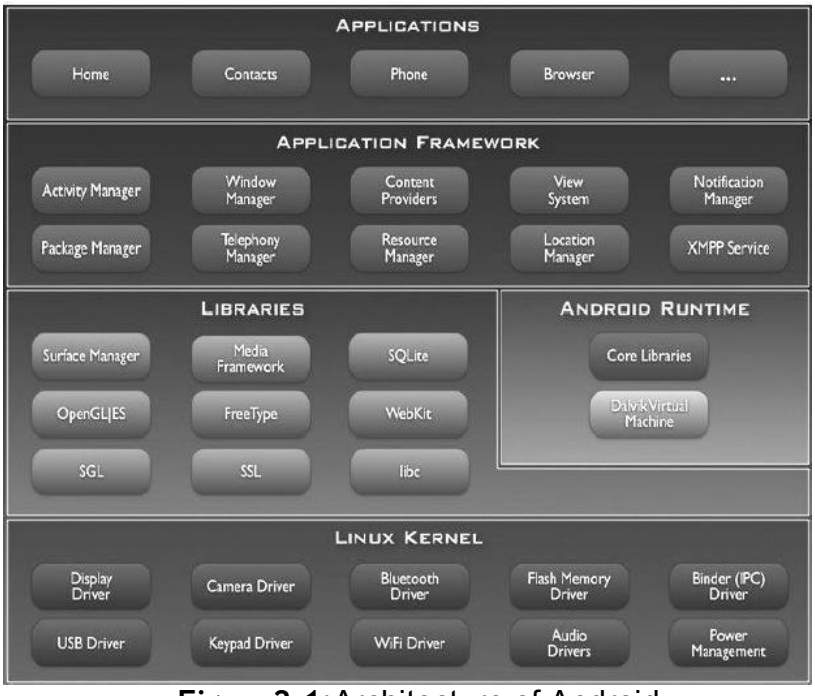


Figure2.1: Architecture of Android

2.2.1. Linux Kernel

Android Architecture is based on Linux 2.6 kernel. It helps to manage security, memory management, process management, network stackand other important issues. Therefore, the user should bring Linux in his mobile deviceasthe main operating systemandinstall all the drivers required in order to run it. Android provides the support for the Qualcomm MSM7K chipset family. For instance, the current kernel tree supports Qualcomm MSM 7200Achipsets, but in the second half of 2008 we should see mobile devices with stable version Qualcomm MSM 7200,whichincludesmajorfeatures:

- WCDMA/ HSUPA and EGPRS network support
- Bluetooth 1.2and Wi-Fi support
- Digital audio support for mp3and other formats
- Support for Linux and otherthird-party operating systems
- Java hardware acceleration and support for Java applications
- Q camera up to 6.0megapixels
- Gps One–solution for GPS

2. Libraries

In the next level there are a set of native libraries written in C/C++, which are responsible for stable performance of various components. For example, Surface Manager is responsible for composing different drawing surfaces on the mobile screen. It manages the access for different processes to compose2D and 3Dgraphic layers. Open GLES and SGL make a core of graphic libraries and are used accordingly for 3Dand 2Dhardwareacceleration. Moreover, it is possible to use 2Dand 3D graphics in the same application in Android. The media framework was provided by PacketVideo, one of the members of OHA. It gives libraries for playback and recording support for all the major media and static image files. Free Type libraries are used to render all the bitmap and vector fonts. For data storage, Android uses SQ Lite. As mentioned before, it is extra light rational management system, which locates a single file for all operations related to database. Web Kit, the
Same browser used by Apples' Safari, was modified by Android in order to fit better in a small size screens.

3. Android Runtime

At the same level there is Android Runtime, where the main component Dalvik Virtual Machine is located. It was designed specifically for Android running in limited environment, where the limited battery, CPU, memory and data storage are the main issues. Android gives an integrated tool "dx", which converts generated byte code from .jar to .dex file, after this byte code becomes much more efficient to run on the small processors.

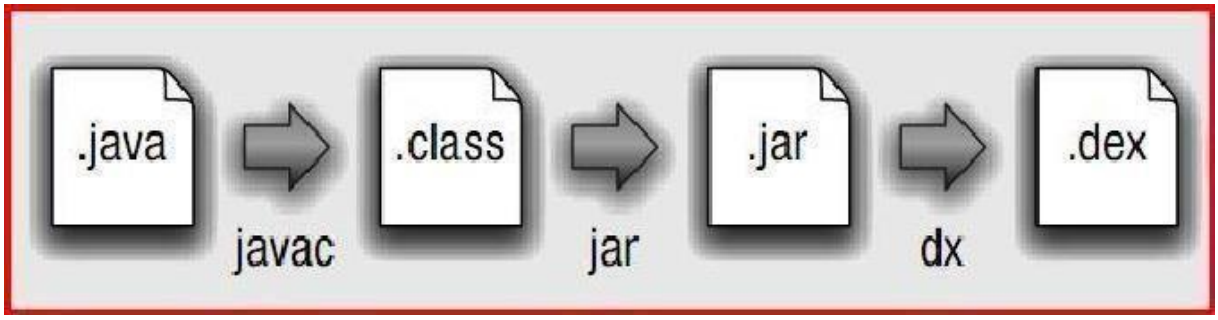


Figure2.2: Conversion from .java to .dexfile

As the result, it is possible to have multiple instances of Dalvik virtual machine running on the single device at the same time. The Core libraries are written in Java language and contains of the collection classes, the utilities, IO and other tools

4. Application Framework

After that, there is Application Framework, written in Java language. It is a toolkit that all applications use, ones which come with mobile device like Contacts or SMS box, or applications written by Google and any Android developer. It has several components. The Activity Manager manages the life circle of the applications and provides common navigation back stack for applications, which are running in different processes. The Package Manager keeps track of the applications, which are installed in the device. The Windows Manager is Java programming language abstraction on the top of lower level services that are provided by the Surface Manager. The Telephony Manager contains of a set of API necessary for calling applications.

5. Application Layer

At the top of Android Architecture we have all the applications, which are used by the final user. By installing different applications, the user can turn his mobile phone into the unique, optimized and smart mobile phone. All applications are written using the Java programming language.

3. DEVELOPINGAPPLICATIONS

1. Application BuildingBlocks

We can think of an Android application as a collection of components, of various kinds. These components are for the most part quite loosely coupled, to the degree where you can accurately describe them as a federation of components rather than a single cohesive application. Generally, these components all run in the same system process. It's possible(and quite common) to create multiple threads within that process, and it's also possible to create completely separate child processes if you need to. Such cases are pretty uncommon though, because Android tries very hard to make processes transparent to your code. Google provides three versions of SDK for Windows, for Mac OSX and one for Linux.
The developer can use Android plug-in for Eclipse IDE or other IDEs such as intel iJ. First step for Android developer is to decompose the prospective application into the components, which are supported by the platform. The major building blocks are these:

- Activity
- Intent Receiver
- Service
- Content Provider

2.3.2. AndroidManifest.xml

The Android Manifest.xml file is the control file that tells the system what to do with all the top-level components(specifically activities, services, intent receivers, and content providers described below) you've created. For instance, this is the "glue "that actually specifies which Intents your Activities receive. A developer should predefine and list all components, which he wants to use in the specific AndroidManifest.xml file. It is a required file for all the applications and is located in the root folder. It is possible to specify all global values for the package, all the components and its classes used, intent filters, which describe where

And when the certain activity should start, permissions and instrumentation like security control and testing

Here is an example of Android Manifest .xml file: 1.<?Xml version="1.0" encoding="utf-8"?>
2.<manifest xml ns:android=http://schemas.android.com/apk/res/android
3.package="dk.mdev.android.hello">
4. <applicationandroid:icon="@drawable/icon">
5. <activity class=".Hello Android"android: label="@string/app_name"> 6.<intent-filter>
7.<action android: value="android.intent.action.MAIN" /> 8.<category android:
value="android.intent.category.LAUNCHER"/>9.</intent-filter>
10.</activity> 11.</application> 12.</manifest>
The line 2 is a name space declaration, which makes a standard Android attributes available for that application. In the line 4 there is a single <application>element, where the developer specifies all application level components and its
properties used by the package. Activity class in the line 5 represents the initial screen the user sees and it may have one or more <intent-filter> elements to describe the actions that activity supports.

2.3.3. Application Lifecycle

In Android, every application runs in its own process, which gives better performance in security, protected memory and other benefits. Therefore, Android is responsible to run and shut down correctly these processes when it is needed. It is important that application developers understand how different application components(in particular Activity, Service, and Broadcast Receiver)impact the lifetime of the application's process. Not using these components correctly can result in the system killing the application's process while it is doing important work.
To determine which processes should be killed when low on memory, Android places each process into an "importance hierarchy" based on the components running in them and the state of those components. These process types are (in order of importance).
1.Aforegroundprocessis onethat is required for what the user is currently doing. Various application components can cause its containing process to be considered foreground in different ways. A process is considered to be in the foreground if any of the following conditions hold:
1. It is running an Activity at the top of the screen that the user is interacting with (it's on Resume() method has been called).
2. It has a Broadcast Receiver that is currentlyrunning(it's Broadcast Receiver. On Receive() method is executing).
3. It has a Servicethat is currently executing code in oneof its callbacks(Service. On Create(), Service. On Start(), or Service. On Destroy()).
4. There will only ever be a few such processes in the system, and these will only be killed as a last resort if memory is so low that not even these processes can continue to run. Generally, at this point, the device has reached a memory paging state, so this action is required in order to keep the user interface responsive.

2. A visible process is one holding an Activity that is visible to the user on-screen but not in the foreground(its on Pause() method has been called).This may occur, for example, if the foreground Activity is displayed as a dialog that allows the previous Activity to be seen behind it. Such a process is considered extremely important and will not be killed unless doing so is required to keep all foreground processes running.

3. A service process is one holding a Service that has been started with the start Service () method. Though these processes are not directly visible to the user, they are generally doing things that the user cares about (such as background mp3 playback or background network data upload or download), so the system will always keep such processes running unless there is not enough memory to retain all foreground and visible process.

3. A background process is one holding an Activity that is not currently visible to the user(its on Stop () method has been called). These processes have no direct impact on the user experience. Provided they implement their Activity life-cycle correctly (see Activity for more details), the system can kill such processes at any time to reclaim memory for one of the three previous processes types. Usually there are many of these processes running, so they are kept in an LRU list to ensure the process that was most recently seen by the user is the last to be killed when running low on memory.

3. An empty process is one that doesn't hold any active application components. The only reason to keep such a process around is as a cache to improve startup time the next time a component of its application needs to run. As such, the system will often kill these processes in order to balance overall system resources between these empty cached processes and the underlying kernel caches.

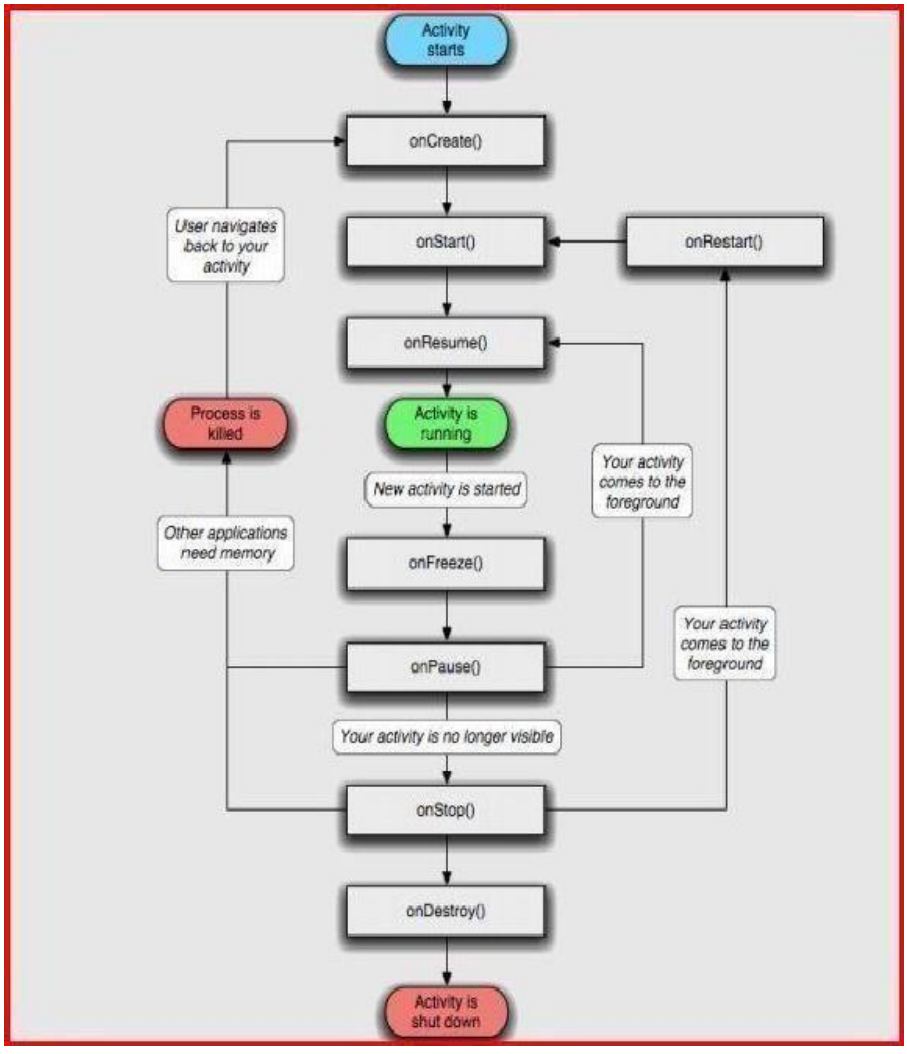
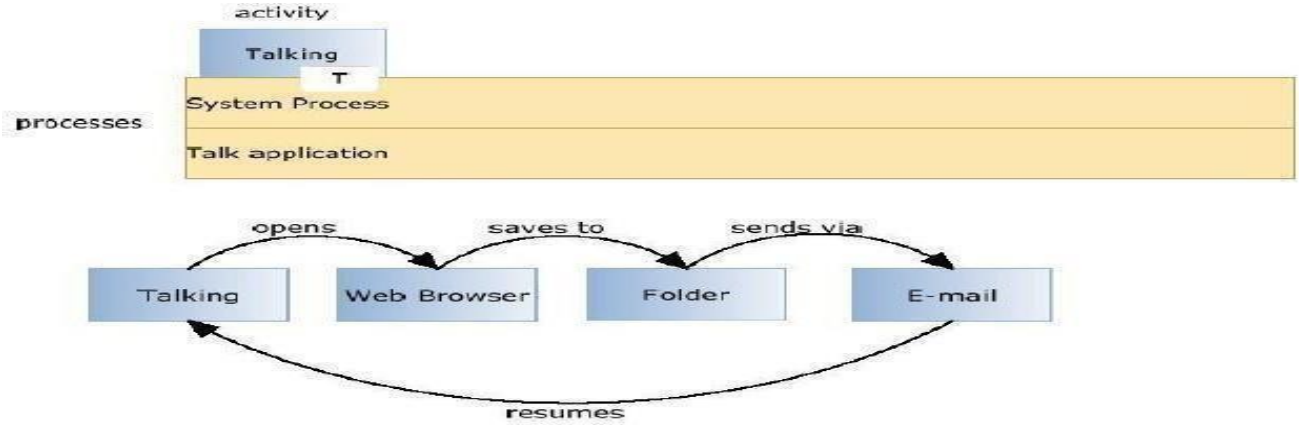
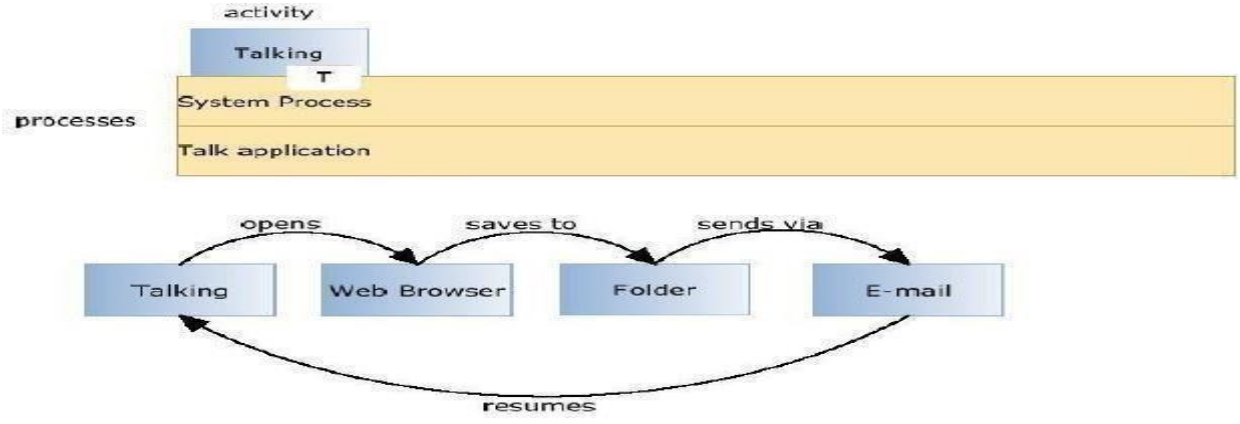


Figure2.3: Flowchart Showing the Life cycle of an Activity

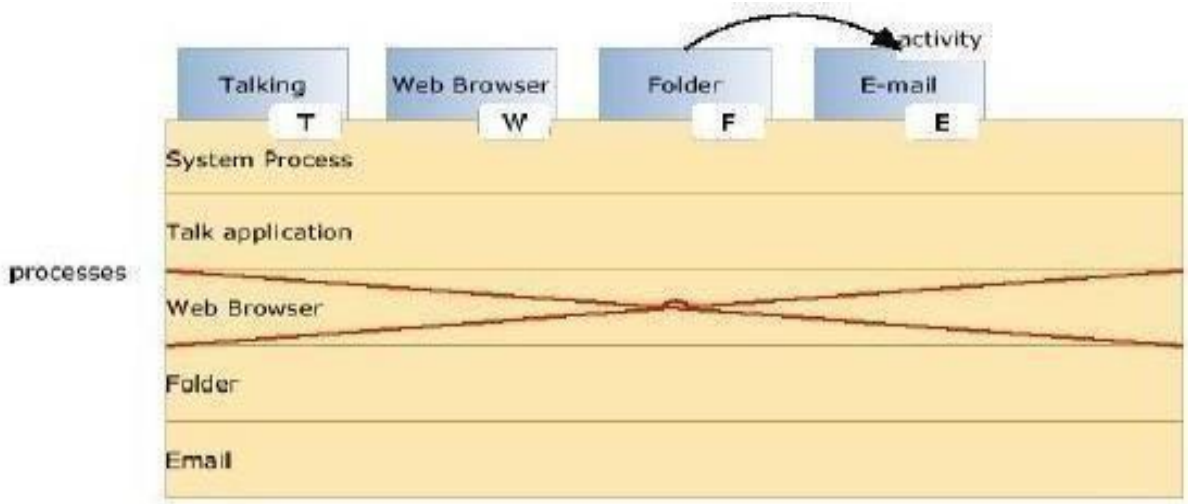
In the following example we will display a process flow from the Android System point of view to get a clear idea how the applications behave. Let assume the



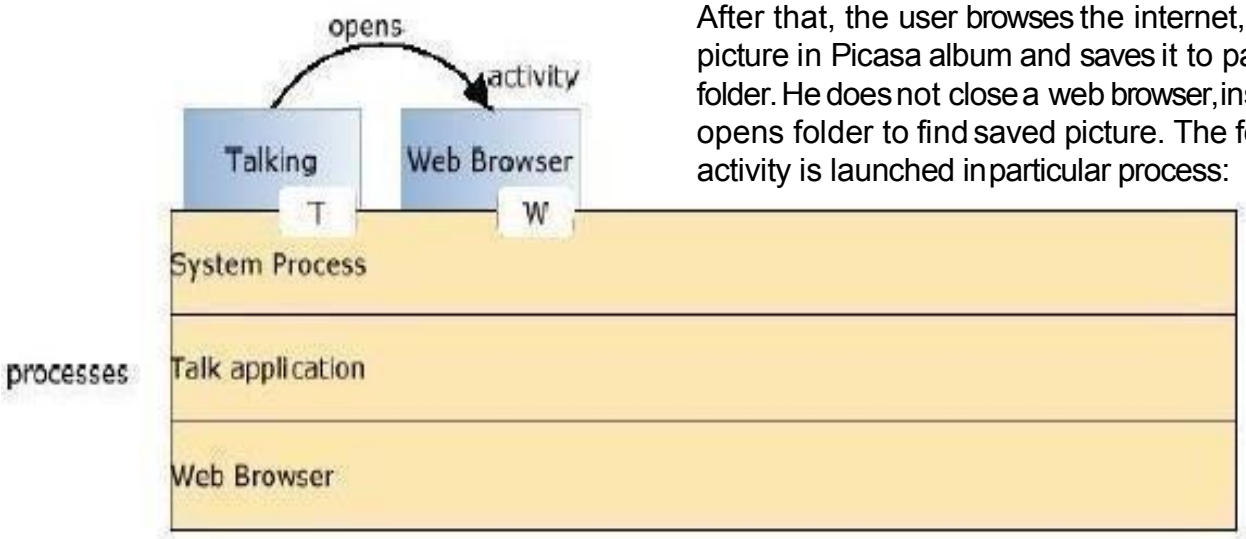
Possible scenario
: A user talks to his friend via mobile phone and he is asked to browse the internet (a talk is hold for a moment), find a picture of him in his Picasa Album, send it via Email back to his friend and resume a talk. In this situation, there are 4 different applications and 4 different processes running, but from the user point of view none of them are important, as Android manages CPU work and memory usage by itself. It means the user can travel through the applications forward and back without thinking about how much memory is left or which processes are run at the time. Firstly, as the user is talking to his friend, a specific Talk application is opened, which contains the activity manager. In the following stack we can see two processes running, the main system process and Talk application process. Moreover, before going to Web Browser application, the system saves a Talk state Tin order to remember that process:

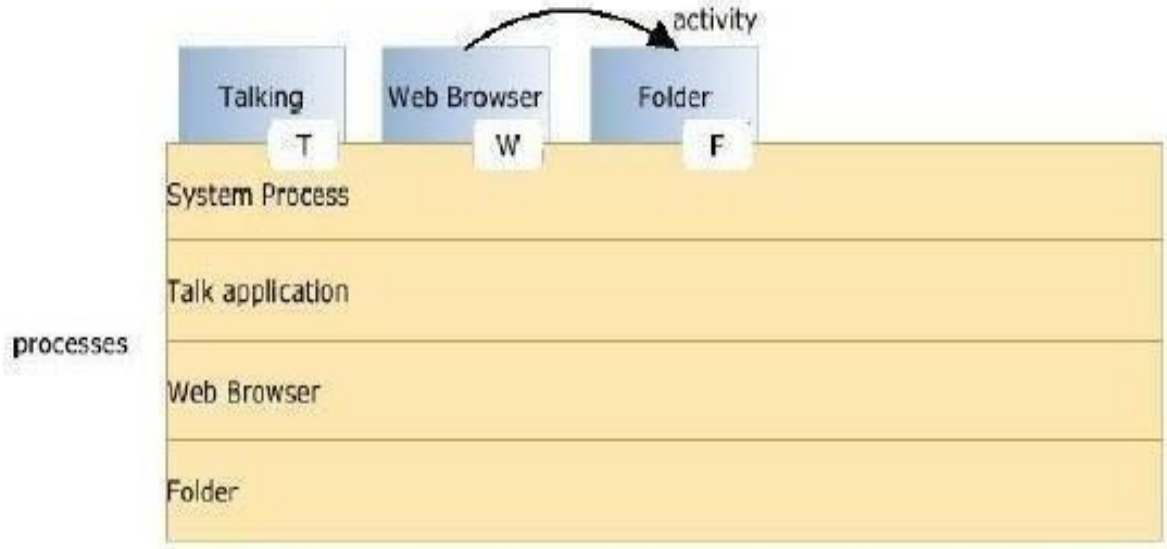


At this point, as a user holds a talk and opens a web browser, the system creates a new process and new web browser activity is launched in it. Again, the state of the last activity is saved(W):



After that, the user browses the internet, finds his picture in Picasa album and saves it to particular folder. He does not close a web browser, instead he opens folder to find saved picture. The folder activity is launched in particular process:





At this point, the user finds his saved picture in the folder and he creates a request to open an Email application. The last state F is saved. Now assume that the mobile phone is out of the memory and there is no room to create a new process for Email application. Therefore, Android looks to kill a process. It cannot destroy Folder process, as it was used previously and could be reused again, so it kills Web Browser process as it is not useful anymore and locates a new Email process instead;

Now the user comes back to the Talk application and resumes his talk with his friend. Because of the saved states, going back procedure is fast and useful, because it remembers previous activities and its views. This example shows, that it does not matter how many applications and processes are active or how much available memory is left, Android it manages fast and without a user interaction.

2.3.4. Application Framework

Developers have full access to the same frame work APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user. Underlying all applications is a set of services and systems, including:
1.A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser

1. Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data

1.A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files

1.ANotification Manager that enables all applications to display custom alerts in the status bar

1. An Activity Manager that manages the life cycle of applications and provides a common navigation backtrack

2.3.5. Library

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 System library (lib c), 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, includingMPEG4,H.264,MP3,AAC, AMR, JPG, and PNG
- Surface Manager- manages access to the display subsystem and seamlessly composites 2Dand 3D graphic layers from multiple applications.
- 3D libraries - an implementation based on OpenGLES1.0APIs; the libraries use either hardware 3D acceleration(where available) or the included, highly optimized 3D softwarer asterizer.

4. SOFTWARE DEVELOPMENT

The feed back on developing applications for the Android platform has been mixed. Issues cited include bugs, lack of documentation, inadequate QA. The first publicly available application was the Snake game.

1. Software Development kit

It includes development and debugging tools, a set of libraries, a device emulator, documentation, sample projects, tutorials, and FAQs. Requirements also include Java Development Kit, Apache Ant, and Python 2.2 or later. The only officially supported integrated development environment (IDE) is Eclipse 3.2 or later, through the Android Development Tools Plug-in, but programmers can use command line tools to create, build and debug Android applications.

Partial Listing of Open Handset Alliance Participants

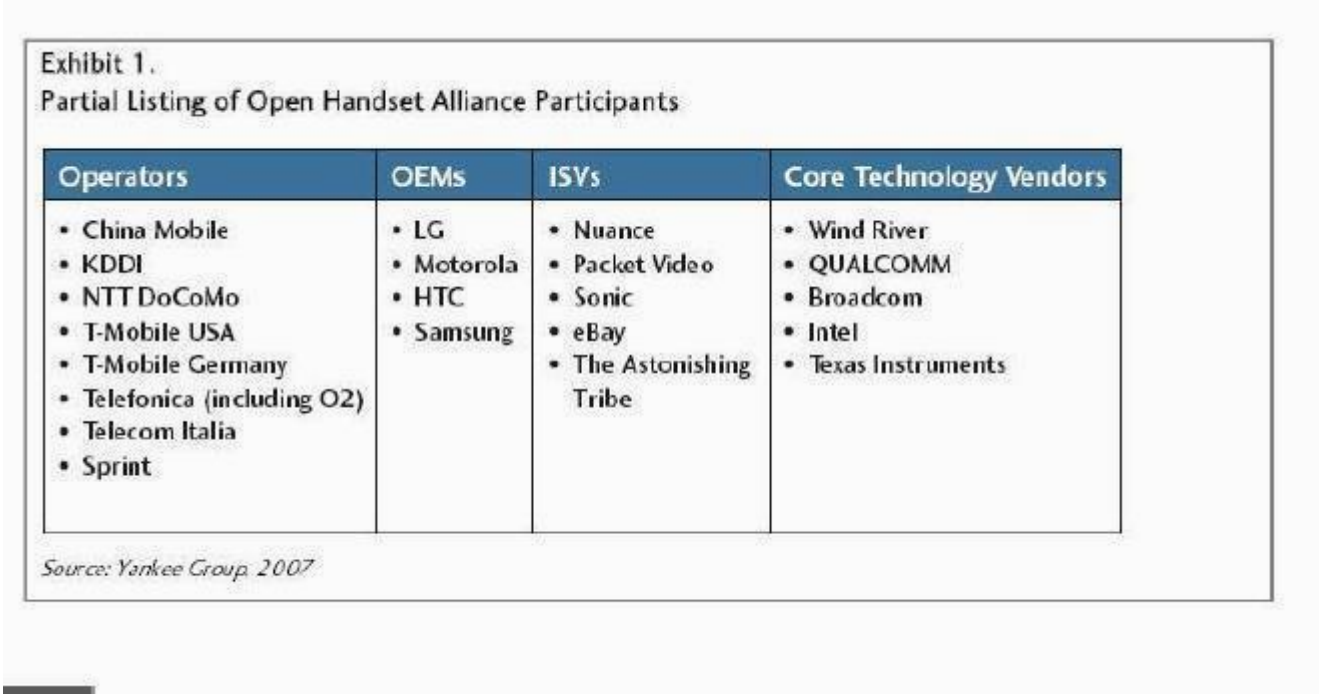


Exhibit 1.
Partial Listing of Open Handset Alliance Participants

Operators	OEMs	ISVs	Core Technology Vendors
• China Mobile • KDDI • NTT DoCoMo • T-Mobile USA • T-Mobile Germany • Telefonica (including O2) • Telecom Italia • Sprint	• LG • Motorola • HTC • Samsung	• Nuance • Packet Video • Sonic • eBay • The Astonishing Tribe	• Wind River • QUALCOMM • Broadcom • Intel • Texas Instruments

Source: Yankee Group 2007

2.5SECURITYISSUES

Android mobile phone platform is going to be more secure than Apple's iPhone or any other device in the long run. There are several solutions nowadays to protect Google phone from various attacks. One of them is security vendor McAfee, member of Linux Mobile (Li Mo) Foundation. This foundation joins particular companies to develop an open mobile - device software platform.

Many of the companies listed in the Li Mo Foundation have also become members of the Open Handset Alliance(OHA). As a result, Linux secure coding practice should successfully be built into the Android Development process. However, open platform has its own disadvantages ,such as source code vulnerability for black-hat hackers. In parallel with great opportunities for mobile application developers, there is an expectation for exploitation and harm. Stealthy Trojans hidden in animated images, particular viruses passed from friend to friend, used for spying and identity theft, all these threats will be active for a long run. Another solution for such attacks is SMobile Systems mobile package.SecurityShield –

an integrated application that includes anti-virus, anti-spam, firewall and other mobile protection is up and ready to run on the Android operating system.

Currently, the main problem is availability for viruses to poseas an application and do things like dial phone numbers, send text messages or multi-media messages or make connections to the Internet during normal device use. It is possible for somebody to use the GPS feature to Track a person's location without their knowledge. Hence SMobile Systems is ready to notify and block these secure alerts. But the truth is that it is not possible to secure your mobile device or personal computer completely, a sit connects to the internet. And neither the Android phone nor other devices will prove to be the exception.

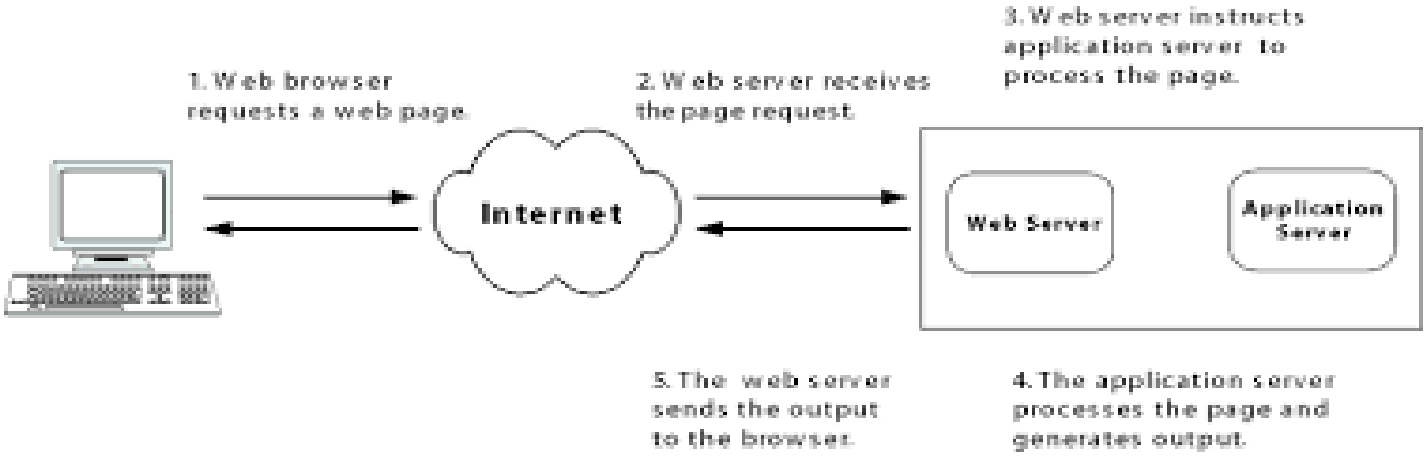
3 . Web Application

Project Design

Before the product implementation begun, a number of designs describing how the software would be developed were made. These covered the product's architecture, its database, how the program would handle its users and its user interface. Only after the initial decisions were made, would development team proceed with the implementation.

3.1 Software architecture

The product's architecture was without a doubt the hardest part to define from the beginning. The team had never written software of this scale. Designing an application like this without previous experience seemed like an overwhelming task. The first section discusses how the team designed the application with regard to a standard client-server



model. The second section discusses how the program's behavior in relation to the user was modeled.

3.1.1 Extending the client-server model

Applications are usually organized by logical parts called "tiers". Each tier is assigned a role. Desktop applications usually have a single tier and reside on the client's computer. Client- server applications on the other hand usually have two or more tiers:

- *The client* – It acts as a presentation tier which interacts with the user
- *The server* – It represents the application tier that manages data and business logic.
- *The database* – Acting as the data tier, storing and retrieving information.

For this project, the application's tiers would have to be expanded, both in functionality and number.

3.1.1.1 Layering application tiers

When the application was first planned, one of the main objectives of the team was to modularize the code. It was the team's intention to allow developers to remove and add "modules" (the main features) without threatening the integrity of the rest of the program. Because the application is divided into several tiers, this new modular system would have to extend over both. Figure 3 shows what it would look like conceptually.

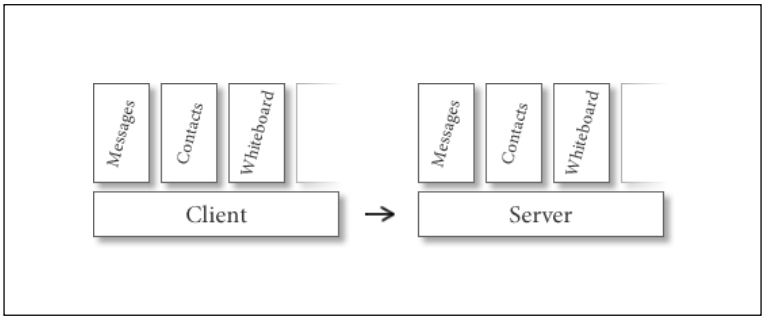


Figure 3 - Client-server model supporting modules

This gave the team the idea to create a basic framework that would:

- Load each module, isolate it and initialize it when the entire program is loaded.
- Handle messaging between the server and the client. It would forward messages appropriately between modules (e.g. the chat module on the client would contact the chat module on the server).

3.1.1.2 Adding an interactivetier

The team realized, as it researched Web Sockets, that the client would need to connect through two channels whereas the original Java client only needed one. This was due to two reasons:

- The application needs to be on the client's computer. Because the Java client is the application, it only needs to download the user's data and use it to populate the

application with it. In the web application's case, the application itself needs to be downloaded, and a web server is needed to do that.

- HTTP connections and socket connections are different. Even in software solutions that merge both functions and can act as both an HTTP Server and a WebSocket Server, two connections have to be established.
  Even if the application is fully loaded on the client's computer with a cache manifest (meaning that the application does not have to connect to the web server again), an HTTP connection has to be made occasionally to check if the manifest is updated.

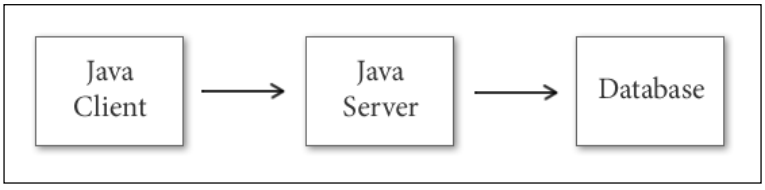


Figure 4 - Standard client-server model with three tiers

In Figure 4, the original client connected to the server and it, in turn, connects to the database. In Figure 5 though, the client connects to two services and they each have to separately connect to the database.

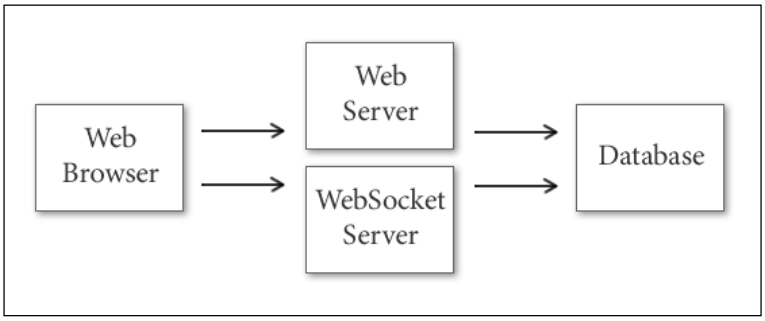


Figure 5 - Client-server model with two connections from the client

The team came to the conclusion that if two connections were used, the application could take advantage of their different behaviors. It could use the HTTP connection to request static pages like forms and read-only content from the database with Ajax and interact with the server through the WebSocket connection.

Because the WebSocket server cannot be changed while it is running, it has to be restarted whenever source code needs to be changed on it. The biggest advantage is that this way, static content is not bound to the WebSocket server and it can be changed on the fly. Figure 6 shows how the applications modular architecture can be merged with a four-tier model.

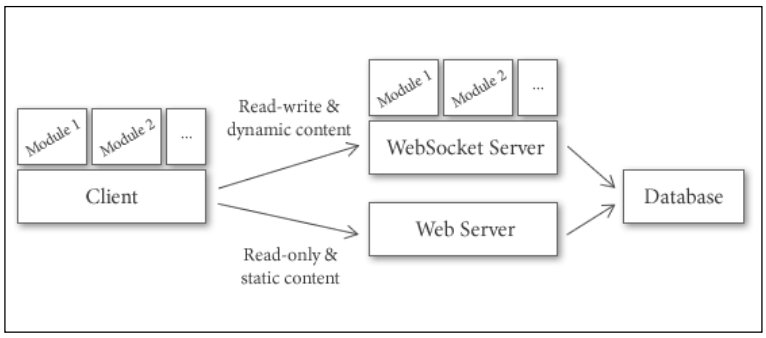


Figure 6 - Division of tasks with four tiers

3.2 The application's database

Since the application data would be stored in a database, it can be stored according to its features and updated through the development process incrementally. Each application feature would have at least one table, and different columns could maintain relationships across tables. Figure 7 displays these relationships and all the fields in detail.

The *users* table would hold a list of user IDs, usernames, e-mails, full names, privilege levels and passwords. The passwords would be stored hashed as insurance if the database is hacked. This way, even if someone recovers the passwords, they would be unreadable.

There would also be a *user To User* table that links users as contacts and gives their relationship a unique ID. The dependence on the user ID field from *users* allows the entry in *user To User* to be deleted if an entry in *users* is deleted effectively making a relationship disappear if one of its members is removed. These kinds of dependencies are present throughout the program.

The *projects* table would assign each project an ID, a title, a description, a link to the owner's ID, a Boolean labeling it as active/inactive and a creation date. An additional table, *project To User*, links projects to their members, to make searching for projects a user belongs to easier after login. It also contains a field to assign a color intended to allow users to color- code their projects.

Messaging would be handled by three tables. The table *user Messages* handles messages from one user to the other, *project Messages* handles messages a user wants to send to all members' inboxes and *posts* handles messages a user wants to leave on the message board. They all link to relevant user and project IDs and have a title, a body and a posting date. They can also be assigned a priority to allow users to highlight important messages.

Two types of calendars would be available: a personal one (*user Calendar*) and one dedicated to projects (*project Calendar*). A user can then organize his personal events and share them with colleagues if he wishes to. It also allows users to set deadlines on projects and share important dates. They both have a title, a description, a start date and duration. The user's

calendar had a field to define it as public if the user wishes to share it and the project calendar has a reference to the poster's ID.

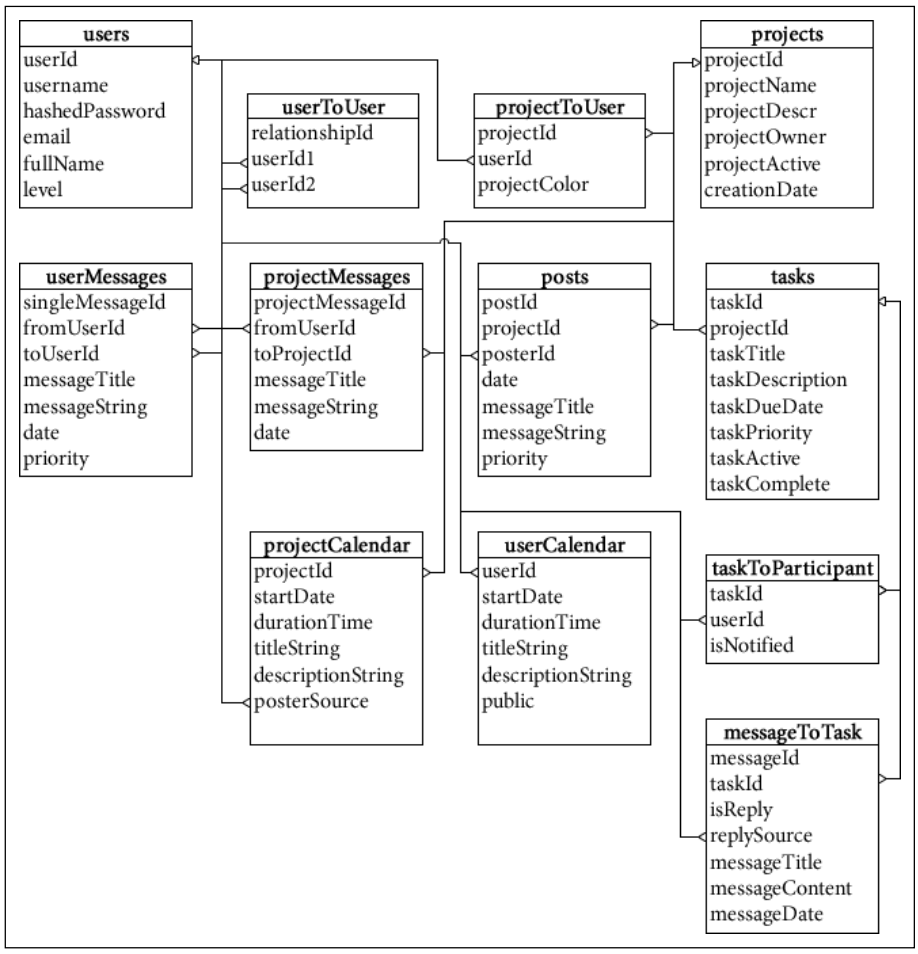


Figure 7 - Relational database model of project

3.3 In-application user interface

Web application developers have to take into consideration that the application will run within the web browser. While it provides plenty of advantages to developers, it also forces certain constraints.

3.3.1 Single document interface

The original Java program used several windows to allow users to see several tasks at once. Since the user is running the program within the web browser, he will most likely handle navigation differently.

Though web browsers can have several windows or tabs open at the same time, it is uncommon for users to navigate between them continuously. When users change windows/tabs, they usually change their work context entirely (e.g. switching from their mail client to their news reader) and seldom need to go back and forth between tabs quickly – unless it is to deal with momentary actions (looking something up quickly, for example). It

is also difficult for different browser windows to communicate, effectively making each window its own program.

A Multiple Document Interface would be ideal to deal with several tasks (as features of this program) simultaneously, but in this case, because the program is inside the web browser, a Single Document Interface is chosen. A diagram of the two solutions is displayed in Figure 9.

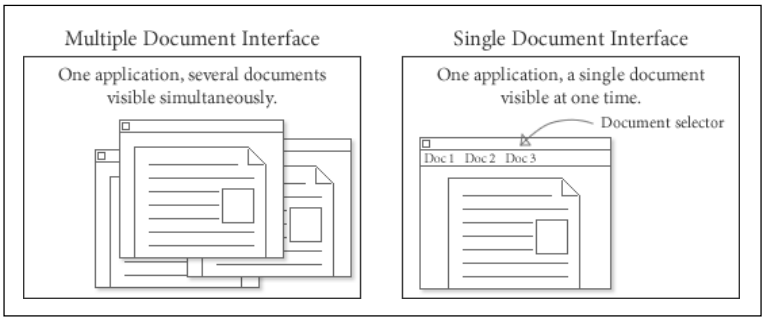


Figure 8 - Multiple and single document interfaces

To make up for the lack of visual responsiveness, the user can be notified of events other ways. If any component needs the user's attention, a small popup can fade in and out somewhere to the edge of the user's field of view where it will not interfere with his work. If the user does not wish to deal with it immediately, a small badge with a counter can be placed above the icon relevant to the notification. Figure 10 shows how this would work.

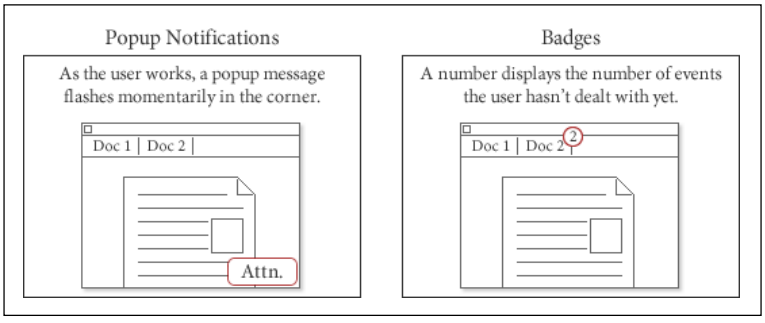


Figure 9 - Popup notifications and badges

3.3.2 Program layout

The program is designed to offer a simple and straightforward interface. The first view the user is displayed is the login screen. If the user is not registered, there is a link to a separate page where he can create a new account.

This web application is designed for the admin side management of the ticket booking system. All the information displayed on the application in a users mobile is the one that is uploaded by the admin and saved in the database through this web app.

The program is designed to offer a simple and straightforward interface. The first view the admin is displayed is the login screen. If the admin is not registered, there is a link to a separate page where he can create a new account.

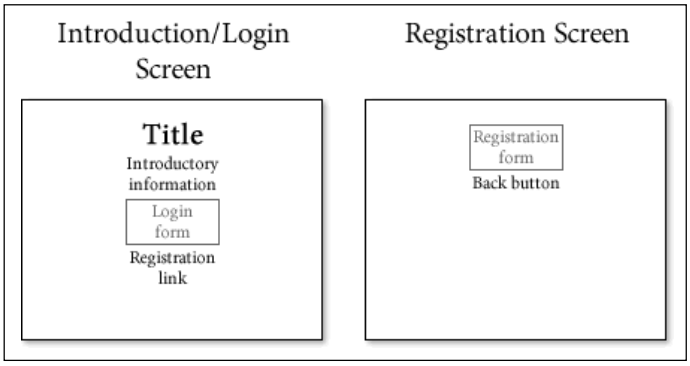


Figure 10 - Simple login screen

Once logged in, the main interface to the program is displayed. A consistent way to display all the features/functions had to be found. It was decided that the program would have a toolbar displaying all the application's options. Its layout can be seen in figure 12.

The toolbar would first display buttons to change between functions on the left. The chosen function would have its title presented in the middle of the toolbar to make the user aware of where he is and the functions content would be beneath the toolbar. On the right-hand side of the toolbar the user's full name is displayed. This gives the current viewer immediate information of what is logged in. If the user's name is clicked, two options are displayed:

- *User settings*: This option takes the user to personal settings that allow the user to change his display name, e-mail, and password and provide a profile picture.
- *Log out*: This option allows the user to end his session and log out of the application.

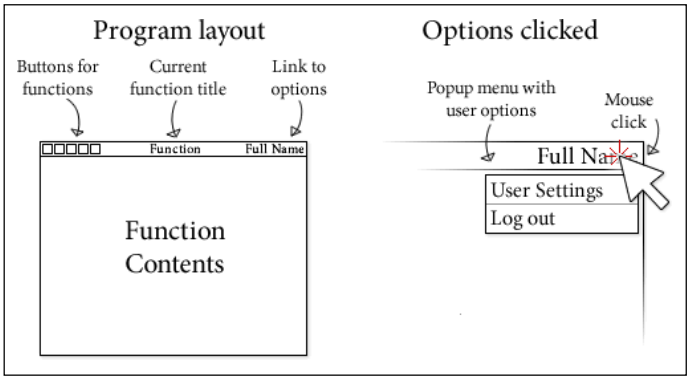


Figure 11 - Program layout and options clicked

BIBLIOGRAPHY

1. https://developer.android.com/
2. https://developers.google.com/training/courses/android-fundamentals
3. https://hackr.io/tutorials/learn-android-development
4. https://en.wikipedia.org/wiki/Android_Studio
5. https://www.tutorialspoint.com/android/index.htm
6. https://en.wikipedia.org/wiki/Android_(operating_system)
7. https://www.comentum.com/web-application-development-process.html
8. https://en.wikipedia.org/wiki/Web_application
9. https://www.maxcdn.com/one/visual-glossary/web-application/
10. https://www.scnsoft.com/blog/web-application-framework