MINISTRY OF DEVELOPMENT OF INFORMATION TECHNOLOGIES AND COMMUNICATIONS OF THE REPUBLIC OF UZBEKISTAN

TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES NAMED AFTER MUHAMMAD AL-KHWARIZMI

Head of the Department

«___» __________ 2017

GRADUATION WORK

Theme: «Development of a mobile application for learning programming languages»

Graduate	______________	Mukhammadiev Sh. K.
	(signature)	
Supervisor	______________	Sharipov Sh.
	(signature)	
Life Safety consultant	______________	
	(signature)	
Reviewer	______________	
	(signature)	

TASHKENT – 2018

MINISTRY OF DEVELOPMENT OF INFORMATION TECHNOLOGIES AND COMMUNICATIONS OF THE REPUBLIC OF UZBEKISTAN

TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES NAMED AFTER MUHAMMAD AL-KHWARIZMI

Faculty: Software engineering Department: System and application programming
Specialty: 5330600 –«Software engineering»

APPROVED BY:
Head of the Department ________________
« ___ »_______________ 2018

ASSIGNMENT

On graduation work of student: Mukhammadiev Sherzod Kahramon o'g'li

1. Theme of work: Development of a mobile application for learning programming languages.
2. Theme approved by University order № 1313.15 from 21.11.2017y.
3. Deadline for completing graduation work: 22.05.2018y.
4. Baseline data for graduation work: Internet resources, theme manuals.
5. Contents of the explanatory note (list of issues to be developed): The analysis of subject area, analysis of the requirements for the development of the system, design of the database structure, creation of information systems.
6. List of graphic material: tables, diagrams, user interfaces, presentation.
7. Date of issue of the assignment: 06.12.2017y.

Supervisor: __________________
(signature)

Task was accepted: __________________
(signature)

8. Consultants for selected sections of the graduation work:

Section	Full name of supervisor	Signature and date	
		Task issued	Task received
Main part	Sharipov Sh.	06.12.2017y.	06.12.2017y.
Life safety	Axmedova.	19.05.2018y.	19.05.2018y.

9. Progress chart:

№	Sections of Graduation Work	Deadline	Mark of supervisor
1	Introduction	10.02.2018	
2	An increase of significance of programming	25.03.2018	
3	Implementation of modern technologies for developing Android apps	10.04.2018	
4	Development of “Dasturlash.net” application	18.05.2018	
5	Life Safety	23.05.2018	
6	Conclusion	30.05.2018	

Graduate: ______________________________«___» __________________ 2018.
(signature)

Supervisor : ____________________________«___» __________________ 2018.
(signature)

Ushbu bitiruv ishi “Dasturlash.net” saytining mobil ilovasini yaratishga bag’ishlangan bo’lib, turli dasturlash tillarini online o’rganish imkoniyatini beradi. Dastur barcha dasturlash tillari va texnologiyalariga oid eng so’nggi maqolalar bilan boyitilgan, shuningdek, elektron darsliklarning barchasi o’zbek tilida. Bu mobil ilova Java dasturlash tili va Sqlite ma’lumotlar bazasi asosida ishlab chiqildi.

Эта выпускная работа посвящена разработке мобильного приложения на веб-сайте Dasturlash.net, и он дает выбор онлайн-обучения различным языкам программирования. Программное обеспечение включает в себя последние статьи, связанные со всеми языками и технологиями программирования, а также все электронные учебники на узбекском языке. Это мобильное приложение создается через язык программирования Java и систему управления базами данных Sqlite.

This final work is dedicated for development of mobile application of ‘Dasturlash.net’ web-site, and it gives choices of online learning different programming languages. The software includes the latest articles related to all programming languages and technologies, as well as, the whole electronic tutorials are in Uzbek. This mobile application is produced via Java programming language and Sqlite database management system.

Table of Contents

INTRODUCTION

From the first days of independence Uzbekistan has paid great attention to the comprehensive development of information and communication technologies and their wide application in all spheres of life of state and society.

Over a short period the authorities created the legal framework conducive to further formation and progress of market of IT-technologies.

In particular, in 1992 the Law "On telecommunications" was adopted, which established general principles of progressive promotion of the industry. Issues of ICT legal regulation received further development in the Law "On information" dated December 11, 2003. President's Resolutions "On measures for further implementation and development of modern information and communication technologies" dated March 21, 2012 and "On further development of computerization and introduction of information and communication technologies" dated May 30, 2002, became important documents in this direction.

At present the Complex program of development of National information and communication system of the Republic of Uzbekistan for 2013-2020 is being implemented. This program was approved by President's Resolution dated June 27, 2013.

Making a statement at enlarged meeting of the Cabinet of Ministers dedicated to the socio-economic development in 2015 and the most important priorities of economic program for 2016, President Islam Karimov noted that in today's conditions in the era of Internet and electronics, the widespread introduction of information and communication technologies in the fields of economy, radical acceleration of creation of system "Electronic government" are of priority significance. As the head of our state underlined, development of ICT has a direct impact on the level of competitiveness of the country, allows you to collect and summarize vast amounts of information, offers great opportunities for management at strategic level.

The task of regular improving of the governance, strengthening the capacity of IT-industry was entrusted to the Ministry for development of information technologies and communications, created by the Resolution of the Head of our state dated February 4, 2015. In addition, since 2002 a Centre for development and introduction of computer and information technologies UZINFOCOM operates, which assists in the development and implementation of national programs of computerization and introduction of ICT in all sectors of public administration, economic and social spheres.

Over the past years, the country carried out systematic work on development of Internet, mobile communications and other areas, on formation of high-tech base of modernization of national economy. The basis for development of ICT in Uzbekistan is the telecommunications infrastructure.

The current stage of development of telecommunications technologies, networks and communication infrastructure of the country is characterized by expansion of fixed and mobile broadband access, increase of switching centers for data transfer and voice traffic, modernization of trunk lines, as well as creation of infrastructure for development of multimedia services.

Over the past 20 years in many regions of the country more than 2,000 kilometers of fiber-optic cables have been laid. They are designed for broadband access to modern technology and provision of converged services such as video telephony, high-speed Internet, watching HDTV-channels and others. Due to the measures taken in 2015, the overall rate of use of international information networks increased by 42.3% compared to 2014 and amounted to 15.5 Gb/s.

Today, all mobile operators operating in our country, consistently introduce the fourth generation network 4G LTE, which allows to handle a large volume of information on Internet quickly and efficiently, download and view video streaming and high-quality photos, use online applications in education purposes and for business. All of these technologies enable Internet users in Uzbekistan to expand their usual ability to work with ICT.

In 2014-2015 Program of development of broadband access networks on Wi-Fi technology has been implemented successfully in the Republic of Uzbekistan. As a result of comprehensive measures at airports, railway stations, places of frequent-stay travelers, parks, shopping malls and other public places of the capital and each administrative center of republic Wi-Fi points have been created.

The high development rates of national Internet segment should be separately noted. Uzbekistan has 10.2 million web users. According to UZINFOCOM center, in January 2016 the number of websites in the UZ zone exceeded 25 thousand, while growth totaled more than 30% compared to same period of last year.

The use of ICT and software products in the management and production processes plays a major role in the development of sectors of the economy and the domestic industry. For instance, in 2014-2015 in the framework of a special state program 86 projects have been realized in order to introduce information systems in large joint-stock companies, associations and organizations totaling more than 330 billion soums.

Particular attention is paid to development of national market of software products.

In order to stimulate domestic programmers the National register of software developers has been created, which already included 69 companies. A directory of software manufacturers Software.uz has been developed that provides necessary information to citizens and businesses.

According to the Resolution of the President of the Republic of Uzbekistan "On measures to further strengthen the incentives of domestic software developers" dated September 20, 2013 new benefits and preferences for members of software industry were introduced. Thus, they are exempt from customs duties for imported equipment for their own use, components, parts, technical documentation and software until January 1, 2017.

It is known that interactive public services are of particular importance in protection of human rights and freedoms, saving time and expenditures for obtaining necessary information and services.

A consistent work on formation of "Electronic government" is carried out in the country. The activity of the Governmental portal of the Republic of Uzbekistan (gov.uz) and the The single portal of interactive state services (SPISS), located on the Internet at my.gov.uz, has been established.

Functional of SPISS expands dynamically, 235 kinds of interactive services are being rendered through it. Over the past five years this system received in total more than 200 thousand electronic applications of citizens and businessmen. Making an online appointment with the heads of government agencies, receipt of information on their activities, various inquiries and sending requests became popular. In January 2015, the portal has launched a new system for discussion of draft legal acts related to business activities, and evaluation of existing documents. To date, 80 draft laws have been discussed through this system, 9 of them have been improved taking into account the opinions of citizens. At this time, the discussion of more than 20 legal acts continues.

Information system E-Sud for electronic proceedings is functioning effectively since 2004. Through its implementation, procedures such as keeping registration books, document management within court, direction of judicial notifications and procedural acts, familiarization of sides with case are completely automated now.

All educational institutions of the republic are connected to ZiyoNET network, which is functioning since 2005. In the library of portal, which was updated in 2014, has more than 75 thousand units of informative-educational resources, including textbooks, dissertations, research papers and others.

As part of implementation of resolution of Head of our state "On measures on further improvement of foreign language learning system" dated December 10, 2012, "Foreign Languages" section has been created on ZiyoNET, which includes over four thousand materials such as textbooks, interactive lessons, games, relevant video and audio. The country regularly hosts major events dedicated to the development of hi-tech industry.

In particular, Week of information and communication technologies ICTWeek Uzbekistan is being held since 2004. Traditionally it is opened with national exhibition of information technologies ICTExpo, which takes place once in two cities - Tashkent and Samarkand. The exhibition presents existing and future forms of ICT-based services, oriented to business community and authorities, and general population. Among the important events of the week - The Forum for Information and Communication Technologies ICTForum, wh ere representatives of leading companies, industry experts and foreign experts discuss state and prospects of progress in this sphere.

As part of the week conferences BestSoft Uzbekistan are also being held, during which they demonstrate the latest achievements of software developers, and e-Government Uzbekistan dedicated to the strategic objectives in the field of "e-government", results of implemented projects, exchange of experience and ideas in this direction. Training of personnel in the development of ICT sector is topical.

Currently, major domestic centers of integration of education, science and industry - Tashkent State Technical University (TSTU) and Tashkent University of Information Technologies (TUIT) - train specialists in technical direction and for IT sector. In 2013, TUIT opened two new master's direction - management of the system "Electronic Government", and library science. At the same time, these universities carry out scientific research on the basis of active cooperation with leading industrial enterprises of the country.

From October 1, 2014 the branch of the prestigious South Korean Inha University began its activity in Tashkent. Professionals in areas such as computer and software engineering, computer network engineering are trained here.

Thus, the domestic IT-industry is developing successfully, joint ventures are being created, new software projects are being developed and implemented, Internet is gaining more space. Ongoing consistent measures in this direction contribute to further development of information society in Uzbekistan and its integration into the global information space.

The importance of the final work. Modern technologies have turned into essential part of our life and it is enough to spin out our head. Thus, there is a need for ideal software to control those technologies.

As a result, learning how to code in different programming languages is so vital so as to develop web-based, mobile and desktop applications.

The main goal of the final work is

.

I. CHAPTER AN INCREASE OF SIGNIFICANCE OF PROGRAMMING

1.1 Today’s insist on software products

The credit of enriching the lives of every individual user goes to mobile applications that have altogether brought a drastic transformation globally. At the same time, it has provided ample scope to the developers to showcase their hidden talent and creativeness. The mobile app technology has taken a curvature that is beyond imagination.

In the present time, mobile apps are reigning the business world in a way that the revenue earned is skyrocketing with each passing day. From the studies it has been found that the total number of downloads in the year 2016 was 149.3 billion and it is expected to go up to 352.9 billion by 2021. “People have become addicted to mobile apps” – is there anything wrong with this statement? Probably NO. People spend 9% more time browsing the apps in comparison to websites. Moreover, users usually spend 2.3 hours on an average, daily navigating the various mobile apps.

REASONS THAT EXPLAIN THE INCREASING
DEMAND OF MOBILE APP DEVELOPMENT

Figure 1. Importance of Mobile Application Development

All the facts and figures mentioned above clearly lay emphasis on the fact that app development industry is enjoying its golden period and the demand for the same is expected to go higher.

Faster & Effective Communication – The present era of mobile apps is enjoying the digitization phase due to which communication has become fast-paced and simpler. One of the primary reasons that has led to better interaction and connection is the increase in a number of social media apps.
As per the recently held survey, Facebook has been identified as the most popular app on both the leading operating systems i.e. Android and iOS. Just like Facebook, other social networking apps have eased out the process of communication, and are the most effective tools for business promotion and advertising. Likewise, WhatsApp is the most used social messaging app with over half a billion people already using it. People have ended up creating business groups on WhatsApp to buy and sell their products.

Demand Of Efficient Developers Has Risen – With the increase in demand for innovative apps both for Android as well as the iOS platforms, the demand for skilled developers have also increased. Most of the app companies are always hunting for efficient app designers who with their caliber and creativity turn out to be the biggest asset of the company. It has been predicted by US Bureau of Labor Statistics that the demand for the number of developers is expected to increase from 17% to 24% by the year 2026.

Developers, get prepared to grab the new job opportunities coming your way! In the upcoming few years, the companies will also look for proficient who expertise in Enterprise apps and IoT apps too.

eCommerce Business Will Reach Heights – The main reason why mobile app development business has gained momentum is that the eCommerce business has become more gigantic. And the number of users each day are growing significantly.

The enormous success of eCommerce business has left the eyes of retail industry wide open. So much so, that the retailers are now looking for app development firms to get an eCommerce app to attract more customers to increase their sales. The popularity of the sites like Amazon, eBay and Flipkart is the live example. Nowadays, more and more people are preferring online shopping for making the purchase because the products are readily available round the clock. As the apps will keep on generating higher revenue for the business, the demand will surge.

Mobile Apps Are A Bliss For The Startups – If you think that mobile applications are only for established companies, then you must venture through the background and success rate of new entrepreneurs and start-ups. The app designing firms are also helping out new business enterprises to set their foot in the highly competitive market. The success of Uber and Airbnb are the live example. There would be hardly few persons who book the cab from the website almost all the customers book a cab for a destination through an app. In the days to come, the demand for mobile applications will ascend.

Addition Of Futuristic Features – Due to the flood of highly advanced devices, the new mobile applications are being developed and introduced in the market at a much higher rate. People nowadays are very smart, so they always keep features on highest priority whilst downloading any app. This is why Google’s Android O and Apple’s iOS 11 have been introduced. WhatsApp is adding new features to enhance the experience of the customers. As the number of devices and features will boom, the demand for app development will undoubtedly flourish.

Budget-Friendly – In the near future, getting apps developed will be both affordable and budget friendly. App building will no longer be considered an item of luxury and will not only meant for large business organizations. We, at Source Soft Solutions, meet the expectations of our clients dedicatedly without pinching their pocket. Moreover, you can get the apps customized and tailor-made. In fact,

building a mobile app is more reasonable than getting a website designed. This again is a proof that the app development business will continue to reach heights.
Enterprise Apps Growth – According to what is going on in the current scenario, with the growth of enterprise apps there will be a positive impact on the high demand for the mobile app. The demand for mobile app development will increase five times by this year ends. As the number of mobile devices are also soaring, the enterprise apps will require the assistance of consumer apps for optimum performance.

The most promising and prospering business is for those companies who have their highly interactive app designed. If you want to maximize profit, and want to take your business to the next level than the need of the hour is to get a mobile app developed necessarily. An investment in mobile app development will help you earn profits in the long run.
With the expert team of mobile app developers at Source Soft Solutions, you should rest assured that your business app will be unique and creative part of your brand.

1.2. The need for distinctive programming languages

We need a programming language in computer programming for the same reason you need a natural language in your everyday life: **to communicate and simplify things.**

How humans interact with each other? using a language , the basic elements of a language are the letters, right? what you do with these letters? you **combine** them if you say h-o-u-s-e , is a nonsense , but if you say house , people will understand you) so what are you doing in reality? you are using a **code** in order to access the world outside, this code is your **interface** on the world.

Consider a deaf-mute person, how he/she communicates? he/she uses a code**,** in this case not the speaking language but the signs language , but for he/she this code has the same function, is an interface on the outside world.

The other function of a code is to simplify things , to give them a structure , to help you build more complex things giving you simpler tools.

Computer programming is problem solving for the most part. Now , imagine that you don’t have any programming language , the only code that you have is the binary system , how much can be practical , to write an entire program , that solve a real problem , using just 0’s and 1’s?

Programming languages are an inteface between humans and computer , they allow us to communicate with computers in order to do awesome things , otherwise will be like having the fastest car in the world but can’t drive it.

I once saw a lecture on MIT OCW , the professor said:

“Programming is the most fun you can have with your clothes on”.

Learning a programming language will help you in several ways.

- Really learn that computers are not magic, they are just very fast logic machines that are very dumb until they are given very detailed instructions (a program). Until you try to teach a computer to do something really simple, you can't really understand how stupid they really are.
- Learn that computers are correct only to the extent that its program is correct, and that it can be hard to make your program absolutely correct. So don't trust computers.
- You will get a better understanding of what is easy for computers (math and looking stuff up and keeping track of things) and what is hard (acting intelligent and following vague instructions).
- If you get good at it, you can get a sense of control over computers. Then you can rule them instead of them ruling you. That's a good feeling.
- It is a valuable skill that could help you get a job.
- It is a valuable skill to understand what programmers need to do in their jobs.

Programming means ‘talking’ to the computer. It means delivering input into it, and expecting some results.

Well, how does one talk to a computer?

I know! He should learn the computer's ‘language’ and then he could talk to it.

But wait!

The ‘language’ of computers is really ugly. Computers recieve input via electronic components that recieve different values.

What should one do, then?

I know! We will have to teach the computer our language, say English, it will take a lot of time (using the method described above), but it's a one time investment that we can get along with. Then we could just talk freely to the computer!

Oh, but wait!

The humans language is too ambiguous. There is the story of the programmer whose wife sent him to the market: “Buy two carrots. If they have cucumbers, buy six.”. He came back with six carrots and said “They had cucumbers”.

Building a computer program (from bare wires!) that will be able to understand the exact meaning of every sentence you say, the same way humans do, is unimaginable. It might be possible with today's technology (Today's languages). Which brings us to the third option.

Why wouldn't we create a new language, that could be easily translated to computer's ‘language’, yet relatively understandable by humans?

This kind of thinking brought us Assembly, which looks like primitive’s men English that lacks some letters.

Then we got C, which has actual English words. It has more advanced syntax that permits shorter, cleaner and safer programs.

Then C++, Python, Java, C#, JavaScript, etc.

Each language gets further from computer's ‘language’. It's OK because with languages like Java it's far more easier to write ‘translators’ for complicated languages than to write them with bare wiring or Assembly.

We need programming languages to ‘talk’ to the computer in a way that is relatively understandable by both sides.

Without a programming language, is available, machine code, bits, values, only numbers, and a program written with this, will work any way, computers can understand easily these codes, programmers, can build applications directly on machine code, and a program can work, but programming language, gives abstraction to make more human readable, the way the programmers do things, at the end, programming languages codes are translated to only machine friendly numerical codes, and program will work, programming in machine code needs a programmer to concern to much, with hardware details, for making the program work, it takes so much preparation, to know all hardware details to make the things work this way, programming languages, releases this requirement making the programmers concern mostly in the program logic, than specifically, hardware details, this makes programming cycle shorter, programming languages include libraries, with functions that minimizes the reinventing the wheel, that is mandatory in machine code, that almost anything should be made starting in zero, if is available, some kind of library, machine codes, to copy and paste in your machine code program, you recognize the need of this characteristic so mature in a programming language, with machine code, there is not an compiler or interpreter, helping you to find errors, in your work, it will take you so much more time, debugging in machine code than using a programming language, and its tools.It is true, that programming languages, are no so optimal translated finally in machine code to same velocity in execution, for the computer to understand, but it is enough, this translating, to argue, the program anyway will work fast enough, even similar velocity, in both cases, a big application will take so much concerns to be made in machine language, the code will be very unreadable, pure numbers, and is possible that with this code programmed this way, the goal of it will be forgotten, so easy, it will be hard to remember what was this code made for, programing languages have these problem anyway, but is a code easier to interpretate, than pure numerics list of orders. Because programming languages give more tools to program, bucles, conditional sentences, variables, etc, a more structured, way, it is interpreted as more power for programing, shorter life cycle to archieve same results, anyway some programming languages, offers some “interfaces”, to work low level code, it is posible, in some languages, to have modules made in assembler, that is like machine code, but a little more readable, is very like machine code, but it have, some names for actions, that the programmer is making. Is recommended use these “interfaces”, for only the code that deppends drastically on speed, a low percentage, of the full program.

Imagine telling a computer to create a red box on the screen for a website. Alright, cool?

In English you would just say:

Create a box.

Sure, ones with straight edges, curled edges?

Make it red.

Wait make the edges red or the whole thing red?

Make it fill the screen.

What screen?

Screen, the thing I’m looking at.

What are you looking at?

Ugh, just forget it.

See the problem now?

Programming languages help convert what we want to do into certain statements that the computer can understand. They are definite. No complications.

Oh you mean why don't we write in binary? Cause that would be frustrating and in the end, someone would assemble(see what I did there) the 1s and 0s and turn it into a programming language. It's all about speeding up coding.

We need a programming language so that we can provide precise instructions for a computer to perform. A programming language exists solely for the benefit of a human programmer; the computer doesn't care what language you use.

A language can be simple and easy to use, like Smalltalk. Or, it can be complicated and difficult to use, like C++. The latter is preferred for high-performance computing, as well as low-level programming. The former is preferred for flexibility and high productivity.

Ultimately, a programming language is used to create the software upon which our digital world relies.

Creating a program directly as a computer will understand without the medium of a programming language/compiler is technically possible.

However, that requires understanding the language of the computer you're working on. With a 64-bit architecture, you will need 64 1's and 0's to do one thing.

0001111000110111001101100011101101100001010110110011001010111010 would be one instruction (this is just a random 64-bit number, but the same length). One line in a programming language will often translate to 2 or 3 instructions at least. A line in a programming language might look like:

```
bob.age = 16;
```

This would require one instruction to write the value 16 to a register, one to find the address of bob.age, and one to write the reg value to the memory location. Meanwhile, those 192 1's and 0's are not nearly as descriptive of their function as the 1 line in the programming language. If you come back to that part of the program, it's much easier to remember why you did something in a descriptive programming language than in 1's and 0's, and it's easier for someone else to understand.

To perform a particular task by computer, programmer prepares a sequence of instructions, know as programmed. A program written for a computer is known as Software. The programmed is stored in RAM .The CPU takes one instruction of the programmed at a time from RAM and executes it. The instructions are executed one by one in sequence and finally produce the desired result.

When the human being stared programming the computer the instruction were given to it in a language that it could easily understand. And that language was machine language i.e. language of 0s and 1s.The writing of programmer in machine language is very cumbersome and complicated and this was accomplished by expert only.

Lots of efforts are made during last 50 years to obviate the difficulties faced for using the machine language. The first language similar to English was developed in 1950 which was known as Assembly Language or Symbolic Programming Languages. After 1960, the High Level Languages were developed which bought the common man very to the computer. And this was the main reason for tremendous growth in computer industry. The high level languages are also known as Procedure Oriented Languages.

The assembly language was easier to use compared with machine la language as it relieved the programmer from a burden of remembering the operation – codes and addresses of memory location. Even though the assembly languages proved to be great help to the programmer, a search was continued for still better languages nearer to the conventional English language. The languages developed which were nearer to the English language, for the use of writing the programmer in 1960 were known as High Level languages. The different high level languages which can be used by the common user are FORTRAN, COBOL, BASIC, PASCAL, PL-1 and many others. Each high level language was developed to fulfill some basic requirements for particular type of problems. But further developments are made in each language to widen its utility for different purposes.

With the invention and popularity of GUI based interfaces. ***GUI based languages are as follows:***

1) Visual basic

2) Visual C++

3) C# (Pronounced as C sharp)

4) Visual Home Page

5) Visual basic 2005

Now we have different programming languages:

- ***PHP:*** This is a server side interpreted non compiled scripting language. It can be written with HTML because the code is executed by the server,the result is displayed to the user as a plain HTML.

- ***JAVA SCRIPT:***It is a client side scripting language. It is embedded in most browser.Objective C:It is object oriented programming languages ,based on C language used by Apple developer.
- ***C++***: C++ is an object oriented programming language used to develop software,video games and more.
- ***JAVA:*** This is a server side interpreted compiled languages,using a virtual machine. It is not a Java Script and is not be related with that.
- ***PYTHON:*** This is a server side interpreted ,open source,non-compiled scripting language. It can be used on its own or as part of other framework like django.
- ***RUBY:*** This is a server side interpreted ,non-compiled scripting language. It is Japanese in origin and no set of specifications. It was released to the public in 1995.
- ***ACTIVE SERVER PAGES(ASAP .NET):*** This is a server side interpreted ,open source,non-compiled scripting language. It is similar to PHP but only run on windows server because it is a Microsoft product in the .NET suite of programming languages.

The need for different languages arises out of the need to do different tasks. Each type of task requires different kinds of repetitive work. This means that if we had a single language, there would be lot's of boilerplate code lying around.

Designing a language to eliminate specific boilerplate paradigms is one reason. Despite all languages being equivalent in expressive power (most languages are Turing complete.[1]), what causes people to turn to different languages is the kind of task they have to do.

A good example is Database management vs systems programming. Both the tasks are achievable via C but Database management code is much more efficient if written in SQL or some other equivalent.

1.3. Targets and achievments of final work

Building mobile applications can be as easy as opening up the IDE, throwing something together, doing a quick bit of testing, and submitting to an App Store – all done in an afternoon. Or it can be an extremely involved process that involves rigorous up-front design, usability testing, QA testing on thousands of devices, a full beta lifecycle, and then deployment a number of different ways. The lifecycle of mobile development is largely no different than the SDLC for web or desktop applications. As with those, there are usually 5 major portions of the process:

1. **Inception** – All apps start with an idea. That idea is usually refined into a solid basis for an application.
2. **Design** – The design phase consists of defining the app's User Experience (UX) such as what the general layout is, how it works, etc., as well as turning that UX into a proper User Interface (UI) design, usually with the help of a graphic designer.
3. **Development** – Usually the most resource intensive phase, this is the actual building of the application.
4. **Stabilization** – When development is far enough along, QA usually begins to test the application and bugs are fixed. Often times an application will go into a limited beta phase in which a wider user audience is given a chance to use it and provide feedback and inform changes.
5. **Deployment**

Often many of these pieces are overlapped, for example, it's common for development to be going on while the UI is being finalized, and it may even inform the UI design. Additionally, an application may be going into a stabilization phase at the same that new features are being added to a new version.

Furthermore, these phases can be used in any number of SDLC methodologies such as Agile, Spiral, Waterfall, etc.

Each of the these phases will be explained in more detail by the following sections.

Inception

The ubiquity and level of interaction people have with mobile devices means that nearly everyone has an idea for a mobile app. Mobile devices open up a whole new way to interact with computing, the web, and even corporate infrastructure.

The inception stage is all about defining and refining the idea for an app. To create a successful app, it's important to ask some fundamental questions. Here are some things to consider before publishing an app in one of the public App Stores:

- **Competitive Advantage** – Are there similar apps out there already? If so, how does this application differentiate from others?

For apps that will be distributed in an Enterprise:

- **Infrastructure Integration** – What existing infrastructure will it integrate with or extend?

Additionally, apps should be evaluated in the context of the mobile form factor:

- **Value** – What value does this app bring users? How will they use it?
- **Form/Mobility** – How will this app work in a mobile form factor? How can I add value using mobile technologies such as location awareness, the camera, etc.?

To help with designing the functionality of an app, it can be useful to define Actors and Use Cases. Actors are roles within an application and are often users. Use cases are typically actions or intents.

For instance, a task tracking application might have two Actors: *User* and *Friend*. A User might *Create a Task*, and *Share a Task* with a Friend. In this case, creating a task and sharing a task are two distinct use cases that, in tandem with the Actors, will inform what screens you’ll need to build, as well as what business entities and logic will need to be developed.

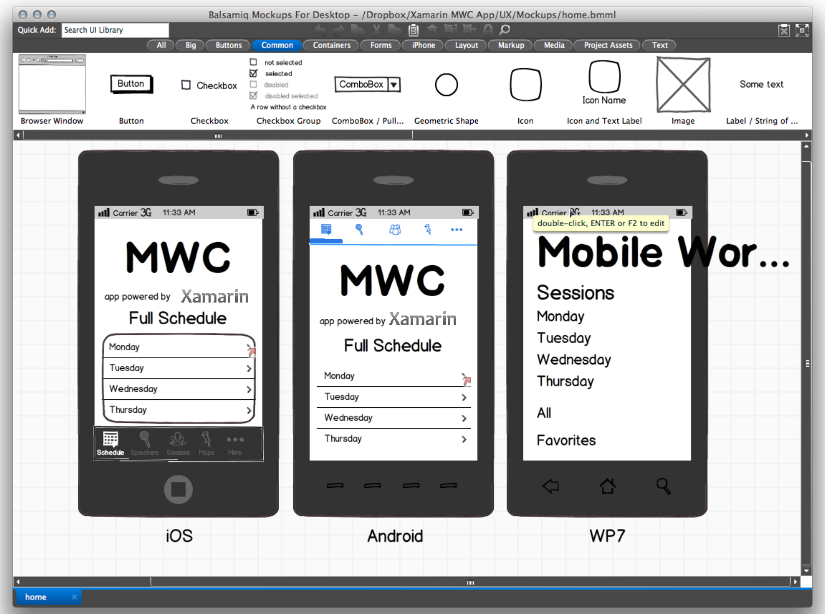
Once an appropriate number of use cases and actors has been captured, it’s much easier to begin designing an application. Development can then focus on how to create the app, rather than what the app is or should do.

Designing Mobile Applications

Once the features and functionality of the app have been determined, the next step is start trying to solve the User Experience or UX.

UX Design

UX is usually done via wireframes or mockups using one of the many design toolkits. UX mockups allow the UX to be designed without having to worry about the actual UI design:



When creating UX mockups, it’s important to consider the interface guidelines for the various platforms that the app will target. The app should "feel at home" on each platform. The official design guidelines for each platform are:

1. **Apple** - Human Interface Guidelines

2. **Android** – Design Guidelines
3. **UWP** – UWP Design basics

II. CHAPTER IMPLEMENTATION OF MODERN TECHNOLOGIES FOR DEVELOPING ANDROID APPS

2.1. Android architecture components

Android architecture components are a collection of libraries that help you design robust, testable, and maintainable apps. Start with classes for managing your UI component lifecycle and handling data persistence.

New lifecycle-aware components help you manage your activity and fragment lifecycles. Survive configuration changes, avoid memory leaks and easily load data into your UI using LiveData, ViewModel, LifecycleObserver and LifecycleOwner.

Avoid boilerplate code and easily convert SQLite table data to Java objects using Room. Room provides compile time checks of SQLite statements and can return RxJava, Flowable and LiveData observables.

The Room persistence library provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite.

The library helps you create a cache of your app's data on a device that's running your app. This cache, which serves as your app's single source of truth, allows users to view a consistent copy of key information within your app, regardless of whether users have an internet connection.

LiveData is an observable data holder class. Unlike a regular observable, LiveData is lifecycle-aware, meaning it respects the lifecycle of other app components, such as activities, fragments, or services. This awareness ensures LiveData only updates app component observers that are in an active lifecycle state.

LiveData considers an observer, which is represented by theObserver class, to be in an active state if its lifecycle is in the STARTED or RESUMED state. LiveData only notifies active observers about updates. Inactive observers registered to watch LiveData objects aren't notified about changes.

You can register an observer paired with an object that implements the LifecycleOwner interface. This relationship allows the observer to be removed when the state of the corresponding Lifecycle object changes to DESTROYED. This is especially useful for activities and fragments because they can safely observe LiveData objects and not worry about leaks—activities and fragments are instantly unsubscribed when their lifecycles are destroyed.

The advantages of using LiveData

Using LiveData provides the following advantages:

Ensures your UI matches your data state

> LiveData follows the observer pattern. LiveData notifies Observer objects when the lifecycle state changes. You can consolidate your code to update the UI in these Observer objects. Instead of updating the UI every time the app data changes, your observer can update the UI every time there's a change.

No memory leaks

> Observers are bound to Lifecycle objects and clean up after themselves when their associated lifecycle is destroyed.

No crashes due to stopped activities

> If the observer's lifecycle is inactive, such as in the case of an activity in the back stack, then it doesn’t receive any LiveData events.

No more manual lifecycle handling

UI components just observe relevant data and don’t stop or resume observation. LiveData automatically manages all of this since it’s aware of the relevant lifecycle status changes while observing.

Always up to date data

If a lifecycle becomes inactive, it receives the latest data upon becoming active again. For example, an activity that was in the background receives the latest data right after it returns to the foreground.

Proper configuration changes

If an activity or fragment is recreated due to a configuration change, like device rotation, it immediately receives the latest available data.

Sharing resources

You can extend a LiveData object using the singleton pattern to wrap system services so that they can be shared in your app. The LiveData object connects to the system service once, and then any observer that needs the resource can just watch the LiveData object. For more information, see Extend LiveData.

The ViewModel class is designed to store and manage UI-related data in a lifecycle conscious way. The ViewModelclass allows data to survive configuration changes such as screen rotations.

The Android framework manages the lifecycles of UI controllers, such as activities and fragments. The framework may decide to destroy or re-create a UI controller in response to certain user actions or device events that are completely out of your control.

If the system destroys or re-creates a UI controller, any transient UI-related data you store in them is lost. For example, your app may include a list of users in one of its activities. When the activity is re-created for a configuration change, the new activity has to re-fetch the list of users. For simple data, the activity can use the onSaveInstanceState() method and restore its data from the bundle in onCreate(), but this approach is only suitable for small amounts of data that can be serialized then deserialized, not for potentially large amounts of data like a list of users or bitmaps.

Another problem is that UI controllers frequently need to make asynchronous calls that may take some time to return. The UI controller needs to manage these calls and ensure the system cleans them up after it's destroyed to avoid potential memory leaks. This management requires a lot of maintenance, and in the case where the object is re-created for a configuration change, it's a waste of resources since the object may have to reissue calls it has already made.

UI controllers such as activities and fragments are primarily intended to display UI data, react to user actions, or handle operating system communication, such as permission requests. Requiring UI controllers to also be responsible for loading data from a database or network adds bloat to the class. Assigning excessive responsibility to UI controllers can result in a single class that tries to handle all of an app's work by itself, instead of delegating work to other classes. Assigning excessive responsibility to the UI controllers in this way also makes testing a lot harder.

It's easier and more efficient to separate out view data ownership from UI controller logic.

2.2. Android object-relational mapping for Sqlite database

For new apps we recommend ObjectBox, a new object-oriented database that is much faster than SQLite and easier to use. For existing apps based on greenDAO we offer DaoCompat for an easy switch.

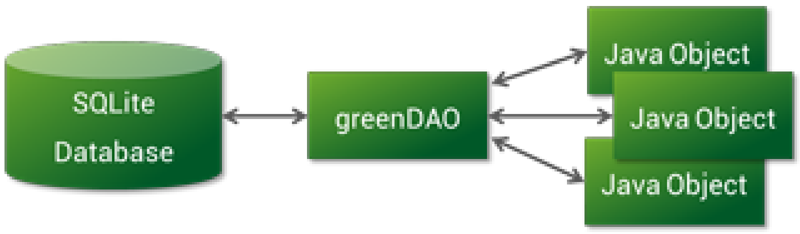


Figure . greenDao technology structure.

GreenDAO is an open source Android ORM making development for SQLite databases fun again. It relieves developers from dealing with low-level database requirements while saving development time. SQLite is an awesome embedded relational database. Still, writing SQL and parsing query results are quite tedious and time-consuming tasks. greenDAO frees you from these by mapping Java objects to database tables (called ORM, “object/relational mapping”). This way you can store, update, delete, and query for Java objects using a simple object oriented API.

greenDAO’s Features at a glance:

1. Maximum performance (probably the fastest ORM for Android); our benchmarks are open sourced too
2. Easy to use powerful APIs covering relations and joins
3. Minimal memory consumption
4. Small library size (<100KB) to keep your build times low and to avoid the 65k method limit
5. Database encryption: greenDAO supports SQLCipher to keep your user’s data safe
6. Strong community: More than 5.000 GitHub stars show there is a strong and active community.
7. Many top Android apps rely on greenDAO. Several of those apps have over 10 million installs.

EventBus is an open-source library for Android and Java using the **publisher/subscriber** pattern for loose coupling. EventBus enables central communication to decoupled classes with just a few lines of code – simplifying the code, removing dependencies, and speeding up app development.

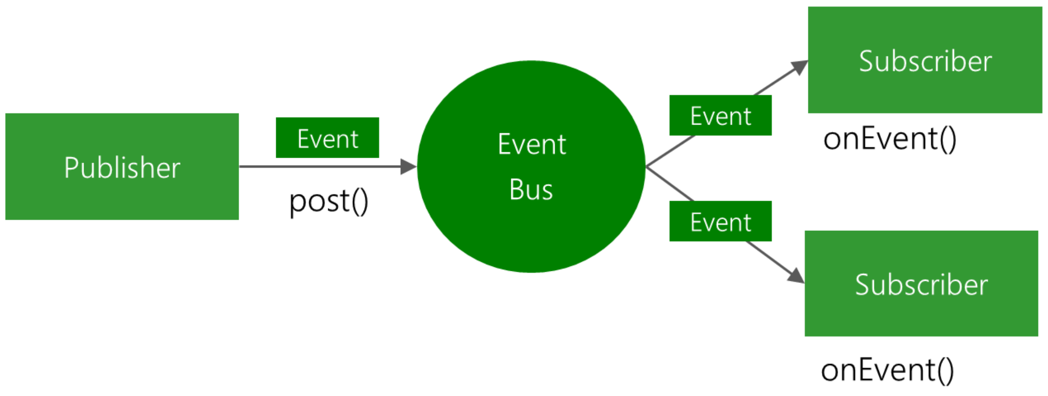


Figure . EventBus technology structure.

Your benefits using EventBus: It…

- simplifies the communication between components
- decouples event senders and receivers
- performs well with UI artifacts (e.g. Activities, Fragments) and background threads
- avoids complex and error-prone dependencies and life cycle issues
- is fast; specifically optimized for high performance
- is tiny (<50k jar)
- is proven in practice by apps with 100,000,000+ installs
- has advanced features like delivery threads, subscriber priorities, etc.

Further EventBus Features

- ➢ **Simple yet powerful:** EventBus is a tiny library with an API that is super easy to learn. Nevertheless, your software architecture may great benefit by decoupling components: Subscribers do not have know about senders, when using events.
- ➢ **Battle tested:** EventBus is one of the most used Android libraries: thousands of apps use EventBus including very popular ones. Over a billion app installs speak for themselves.

- **--High Performance:** Especially on Android, performance matters. EventBus was profiled and optimized a lot; probably making it the fastest solution of its kind.
- **Convenient Annotation based API** (without sacrificing performance)**:** Simply put the @Subscribe annotation to your subscriber methods. Because of a build time indexing of annotations, EventBus does not need to do annotation reflection during your app's run time, which is very slow on Android.
- **Android main thread delivery:** When interacting with the UI, EventBus can deliver events in the main thread regardless how an event was posted.
- **Background thread delivery:** If your subscriber does long running tasks, EventBus can also use background threads to avoid UI blocking.
- **Event & Subscriber inheritance:** In EventBus, the object oriented paradigm apply to event and subscriber classes. Let's say event class A is the superclass of B. Posted events of type B will also be posted to subscribers interested in A. Similarly the inheritance of subscriber classes are considered.
- **Zero configuration:** You can get started immediately using a default EventBus instance available from anywhere in your code.
- **Configurable:** To tweak EventBus to your requirements, you can adjust its behavior using the builder pattern.

2.3. Cloud storage with Firebase on Android

Cloud Storage is built for app developers who need to store and serve user-generated content, such as photos or videos. Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality. You can use our SDKs to store images, audio, video, or other user-generated content. On the server, you can use Google Cloud Storage, to access the same files.

Key capabalities

Robust operations	Firebase SDKs for Cloud Storage perform uploads and downloads regardless of network quality. Uploads and downloads are robust, meaning they restart where they stopped, saving your users time and bandwidth.
Strong security	Firebase SDKs for Cloud Storage integrate with Firebase Authentication to provide simple and intuitive authentication for developers. You can use our declarative security model to allow access based on filename, size, content type, and other metadata.
High scalability	Cloud Storage for Firebase is built for exabyte scale when your app goes viral. Effortlessly grow from prototype to production using the same infrastructure that powers Spotify and Google Photos.

How does it work?

Developers use the Firebase SDKs for Cloud Storage to upload and download files directly from clients. If the network connection is poor, the client is able to retry the operation right where it left off, saving your users time and bandwidth.

Cloud Storage stores your files in a Google Cloud Storage bucket, making them accessible through both Firebase and Google Cloud. This allows you the flexibility to upload and download files from mobile clients via the Firebase SDKs, and do server-side processing such as image filtering or video transcoding usingGoogle Cloud Platform. Cloud Storage scales automatically, meaning that there's no need to migrate to any other provider. Learn more about all the benefits of our integration with Google Cloud Platform.

The Firebase SDKs for Cloud Storage integrate seamlessly with Firebase Authentication to identify users, and we provide a declarative security language that lets you set access controls on individual files or groups of files, so you can make files as public or private as you want.

Implementation path

Integrate the Firebase SDKs for Cloud Storage.	Quickly include clients via Gradle, CocoaPods, or a script include.
Create a	Reference the path to a file, such as

Reference	"images/mountains.png", to upload, download, or delete it.
Upload or Download	Upload or download to native types in memory or on disk.
Secure your Files	Use Firebase Security Rules for Cloud Storage to secure your files.

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably deliver messages at no cost.

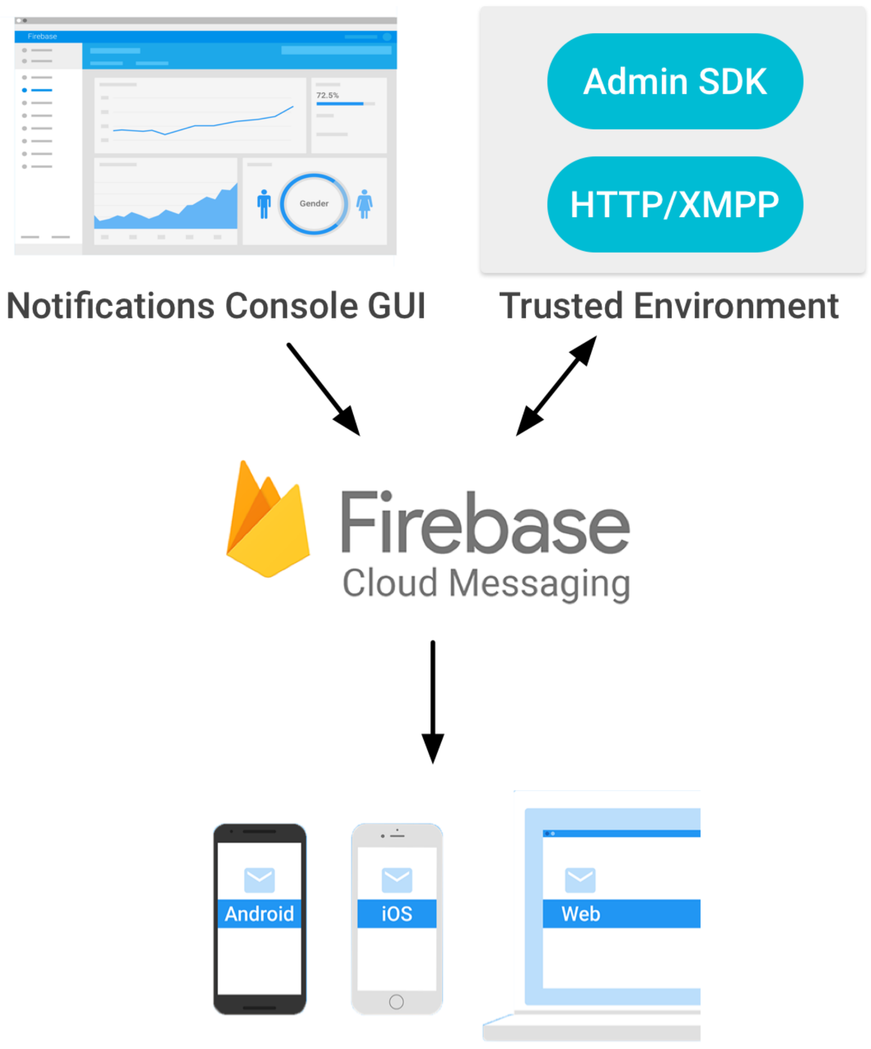
Using FCM, you can notify a client app that new email or other data is available to sync. You can send notification messages to drive user re-engagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4KB to a client app.

Key capabilities

Send notification messages or data messages	Send notification messages that are displayed to your user. Or send data messages and determine completely what happens in your application code. See Message types.

Versatile message targeting	Distribute messages to your client app in any of 3 ways—to single devices, to groups of devices, or to devices subscribed to topics.
Send messages from client apps	Send acknowledgments, chats, and other messages from devices back to your server over FCM’s reliable and battery-efficient connection channel.

How does it work?



An FCM implementation includes two main components for sending and receiving:

1. A trusted environment such as Cloud Functions for Firebase or an app server on which to build, target, and send messages.
2. An iOS, Android, or web (JavaScript) client app that receives messages.

You can send messages via the Admin SDK or theHTTP and XMPP APIs. For testing or for sending marketing or engagement messages with powerful built-in targeting and analytics, you can also use the Notifications composer.

Implementation path

Set up the FCM SDK	Set up Firebase and FCM on your app according to the setup instructions for your platform.
Develop your client app	Add message handling, topic subscription logic, or other optional features to your client app. During the development, you can easily send test messages from the Notifications composer.
Develop your app server	Decide whether you want to use the Admin SDK or one of the server protocols to create your sending logic—logic to authenticate, build sendrequests, handle responses, and so on. Then build out the logic in your trusted environment. Note that if you want to use upstream messaging from your client applications, you must use XMPP, and that Cloud Functions does not support the persistent connection required by XMPP.

2.4. Android networking with Retrofit

Retrofit is a type-safe REST client for Android (or just Java) developed by Square. The library provides a powerful framework for authenticating and interacting with APIs and sending network requests with OkHttp. This library makes downloading JSON or XML data from a web API fairly straightforward. Once the data is downloaded then it is parsed into a Plain Old Java Object (POJO) which must be defined for each "resource" in the response.

Retrofit turns your HTTP API into a Java interface.

```
public interface GitHubService {

  @GET("users/{user}/repos")

  Call<List<Repo>> listRepos(@Path("user") String user);

}
```

The Retrofit class generates an implementation of the GitHubService interface.

```
Retrofit retrofit = new Retrofit.Builder()

    .baseUrl("https://api.github.com/")

    .build();

GitHubService service = retrofit.create(GitHubService.class);
```

Each Call from the created GitHubService can make a synchronous or asynchronous HTTP request to the remote webserver.

```
Call<List<Repo>> repos = service.listRepos("octocat");
```

Use annotations to describe the HTTP request:

- URL parameter replacement and query parameter support
- Object conversion to request body (e.g., JSON, protocol buffers)
- Multipart request body and file upload

API Declaration

Annotations on the interface methods and its parameters indicate how a request will be handled.

REQUEST METHOD

Every method must have an HTTP annotation that provides the request method and relative URL. There are five built-in annotations: GET, POST, PUT, DELETE, and HEAD. The relative URL of the resource is specified in the annotation.

```
@GET("users/list")
```

You can also specify query parameters in the URL.

```
@GET("users/list?sort=desc")
```

URL MANIPULATION

A request URL can be updated dynamically using replacement blocks and parameters on the method. A replacement block is an alphanumeric string surrounded by { and }. A corresponding parameter must be annotated with @Path using the same string.

```
@GET("group/{id}/users")

Call<List<User>> groupList(@Path("id") int groupId);
```

Query parameters can also be added.

```
@GET("group/{id}/users")

Call<List<User>> groupList(@Path("id") int groupId, @Query("sort") String
sort);
```

For complex query parameter combinations a Map can be used.

```
@GET("group/{id}/users")

Call<List<User>> groupList(@Path("id") int groupId, @QueryMap Map<String,
String> options);
```

REQUEST BODY

An object can be specified for use as an HTTP request body with the @Body annotation.

```
@POST("users/new")

Call<User> createUser(@Body User user);
```

The object will also be converted using a converter specified on the Retrofit instance. If no converter is added, only RequestBodycan be used.

FORM ENCODED AND MULTIPART

Methods can also be declared to send form-encoded and multipart data.

Form-encoded data is sent when @FormUrlEncoded is present on the method. Each key-value pair is annotated with @Fieldcontaining the name and the object providing the value.

```
@FormUrlEncoded

@POST("user/edit")

Call<User> updateUser(@Field("first_name") String first, @Field("last_name")
String last);
```

Multipart requests are used when @Multipart is present on the method. Parts are declared using the @Part annotation.

```
@Multipart

@PUT("user/photo")

Call<User> updateUser(@Part("photo") RequestBody photo, @Part("description")
RequestBody description);
```

Multipart parts use one of Retrofit's converters or they can implement RequestBody to handle their own serialization.

HEADER MANIPULATION

You can set static headers for a method using the @Headers annotation.

```
@Headers("Cache-Control: max-age=640000")

@GET("widget/list")

Call<List<Widget>> widgetList();

@Headers({

   "Accept: application/vnd.github.v3.full+json",

   "User-Agent: Retrofit-Sample-App"

})

@GET("users/{username}")

Call<User> getUser(@Path("username") String username);
```

Note that headers do not overwrite each other. All headers with the same name will be included in the request.

A request Header can be updated dynamically using the @Header annotation. A corresponding parameter must be provided to the @Header. If the value is null, the header will be omitted. Otherwise, toString will be called on the value, and the result used.

```
@GET("user")

Call<User> getUser(@Header("Authorization") String authorization)
```

Headers that need to be added to every request can be specified using an OkHttp interceptor.

SYNCHRONOUS VS. ASYNCHRONOUS

Call instances can be executed either synchronously or asynchronously. Each instance can only be used once, but calling clone()will create a new instance that can be used.

On Android, callbacks will be executed on the main thread. On the JVM, callbacks will happen on the same thread that executed the HTTP request.

Retrofit Configuration

Retrofit is the class through which your API interfaces are turned into callable objects. By default, Retrofit will give you sane defaults for your platform but it allows for customization.

CONVERTERS

By default, Retrofit can only deserialize HTTP bodies into OkHttp's ResponseBody type and it can only accept its RequestBody type for @Body.

Converters can be added to support other types. Six sibling modules adapt popular serialization libraries for your convenience.

- Gson: com.squareup.retrofit2:converter-gson
- Jackson: com.squareup.retrofit2:converter-jackson
- Moshi: com.squareup.retrofit2:converter-moshi
- Protobuf: com.squareup.retrofit2:converter-protobuf
- Wire: com.squareup.retrofit2:converter-wire
- Simple XML: com.squareup.retrofit2:converter-simplexml
- Scalars (primitives, boxed, and String): com.squareup.retrofit2:converter-scalars

Here's an example of using the GsonConverterFactory class to generate an implementation of the GitHubService interface which uses Gson for its deserialization.

```
Retrofit retrofit = new Retrofit.Builder()

    .baseUrl("https://api.github.com")

    .addConverterFactory(GsonConverterFactory.create())

    .build();



GitHubService service = retrofit.create(GitHubService.class);
```

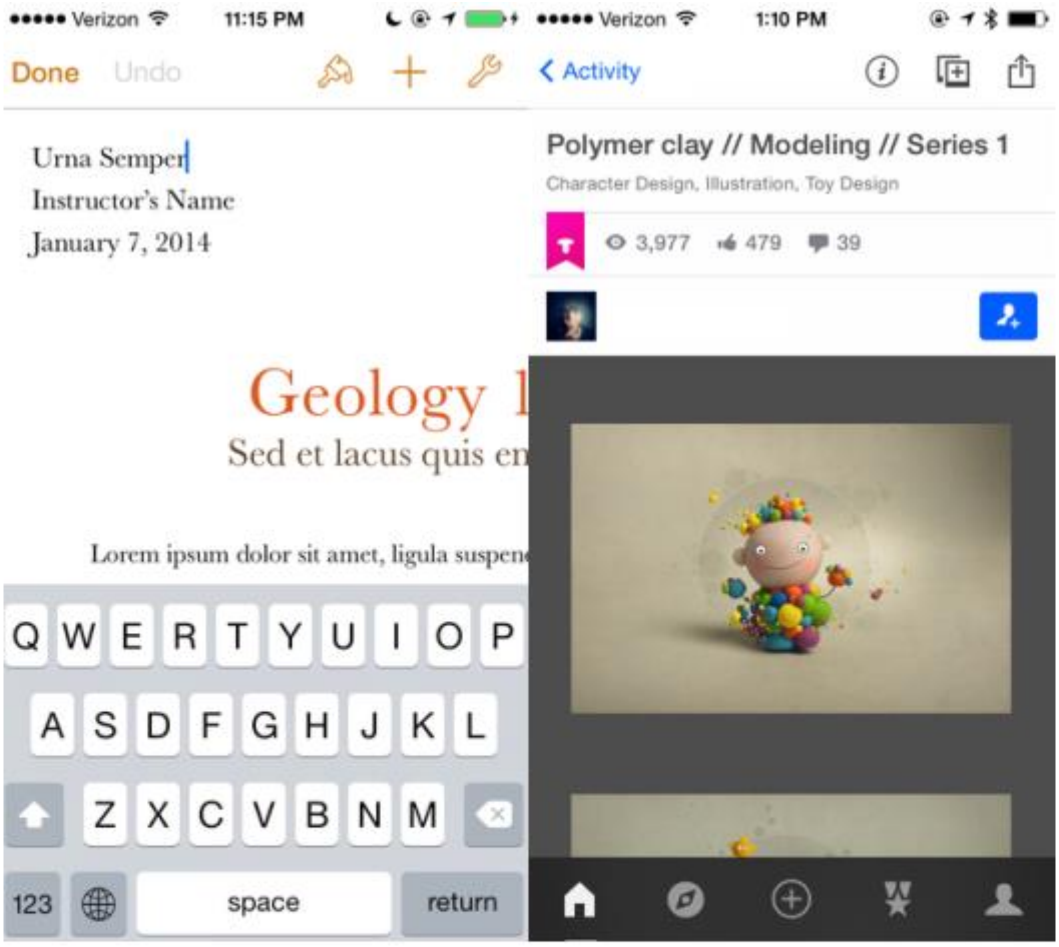
CUSTOM CONVERTERS

If you need to communicate with an API that uses a content-format that Retrofit does not support out of the box (e.g. YAML, txt, custom format) or you wish to use a different library to implement an existing format, you can easily create your own converter. Create a class that extends the Converter.Factory class and pass in an instance when building your adapter.

III. CHAPTER DEVELOPMENT OF "DASTURLASH.NET" APPLICATION

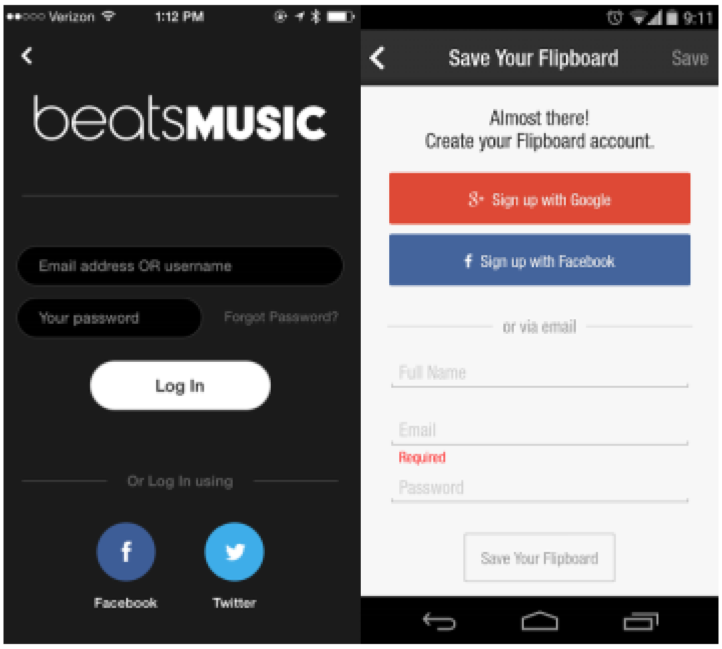
3.1. Software development process

Action bars are so important aspect for all types of software products such as desktop, web and mobile **applications, therefore, implementation of that view is being developed by Google I/O.**



Problem
The user wants quick access to frequently used actions.
Solution
Provide quick access to important actions from the app's action bar (or "toolbar" in
iOS terminology). While navigation bars have dominated web and early mobile application design, the use of other patterns like drawers, slideouts & sidebars, links to
everything, button transformations, vertical and content-based navigation have allowed for more simple app views that can focus on primary and secondary actions,

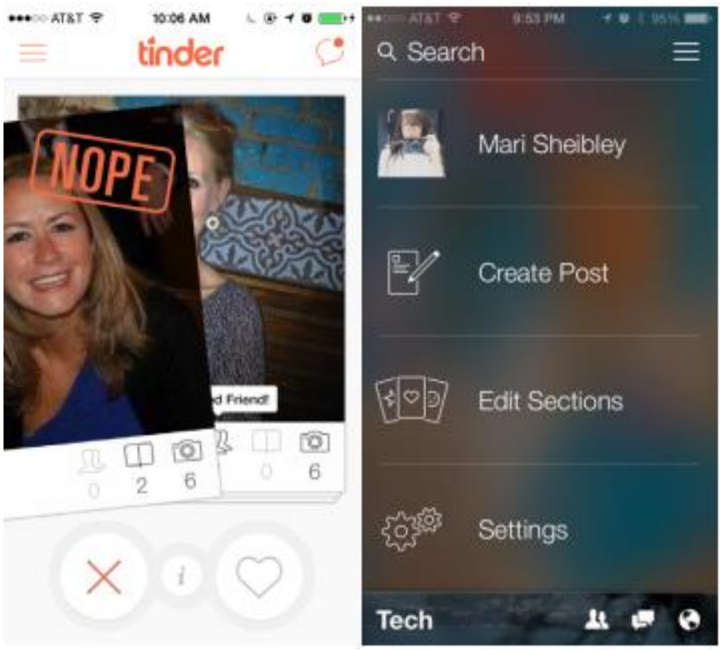
and less on secondary navigation. Common actions are search, share and creating
new content within the app. This persistent menu helps users become familiar with
the UI but also clears away some clutter by focusing on the important actions that
are relevant to the user.



Problem
The user wants an easier way of signing up and logging in.
Solution
Integrate social sign in methods that allow users to login through their existing
accounts. This means they have one less username/password
combination to worry about, and at the same time, you don’t have to worry about password security
as much. Facebook, Twitter and Google are the major OAuth login providers and

depending on the platform and target audience, you can implement all or either of
these in your app instead of having users set up a separate account that they may
or may not end up using in the future. Using this signup and login pattern can also
provide you with some basic data about the user (which feeds into data

auto-population as they use the application), all the while making it easier on them by not
forcing them to type their details into the strange new app they just downloaded.
This simple feature can go a long way in drastically improving your UX, and no wonder this pattern is well on its way to becoming an expectation.



Problem

The user wants to know immediately which actions they can take.

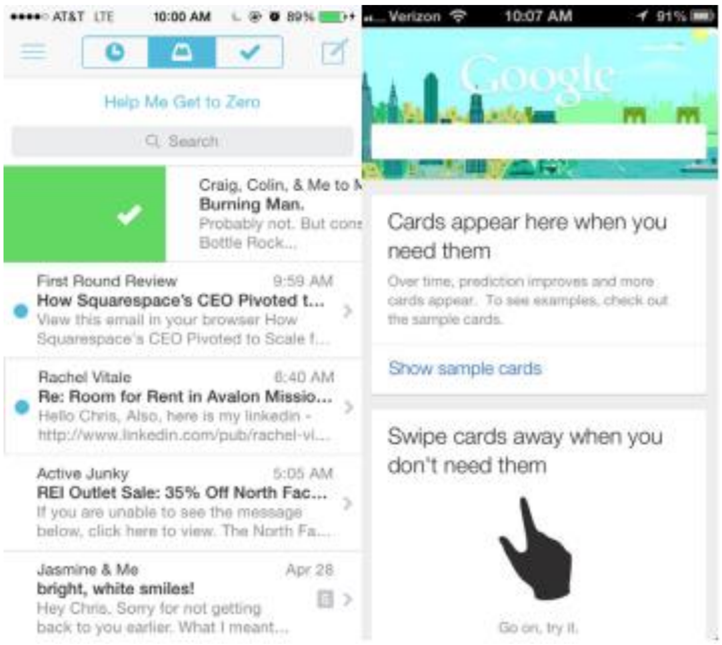
Solution

The ideal touch screen tap target size may be 72px, but some apps like Tinder also give you huge buttons so you know exactly what to do and can do it quickly wherever you are and whatever you're doing - it's pretty hard to miss these massive buttons, even if you're not looking. This is particularly valuable in more simple applications where there are limited actions a user needs to take and, thus, more reason to

31

make it easier for them to take those actions in various contexts. Shazam for example, is meant to be used while watching TV or listening to music, and it really only does one thing. The huge buttons are a great improvement for the user who’s trying to multitask in this distracted state.

3.2. User instruction


AT&T LTE 10:00 AM 89%

Help Me Get to Zero

Search

Craig, Colin, & Me to M
Burning Man.
Probably not. But cons
Bottle Rock...

First Round Review 9:59 AM
How Squarespace's CEO Pivoted t...
View this email in your browser How Squarespace's CEO Pivoted to Scale f...

Rachel Vitale 8:40 AM
Re: Room for Rent in Avalon Missio...
Hello Chris, Also, here is my linkedin - http://www.linkedin.com/pub/rachel-vi...

Active Junky 5:05 AM
REI Outlet Sale: 35% Off North Fac...
If you are unable to see the message below, click here to view. The North Fa...

Jasmine & Me Apr 28
bright, white smiles!
Hey Chris. Sorry for not getting back to you earlier. What I meant...

Verizon 10:07 AM 91%

Google

Cards appear here when you need them

Over time, prediction improves and more cards appear. To see examples, check out the sample cards.

Show sample cards

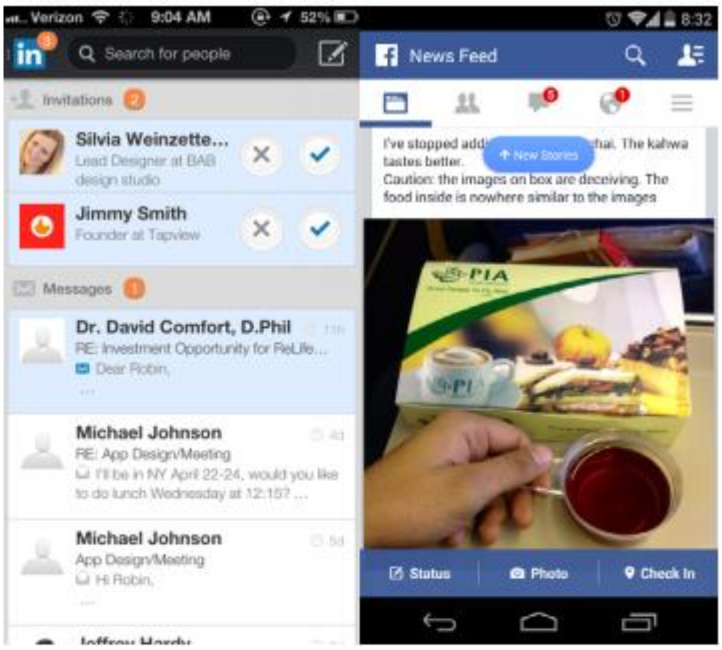
Swipe cards away when you don't need them

Go on, try it.

Problem
The user wants to focus on particular content.
Solution
Allow content to be swiped or moved out of the way. This provides users with a very
intuitive way of handling the information on screen. For example, the “cards” in
Google Now can be swiped away when you don’t need them to clear up the clutter;
similarly, profles in Tinder can be swiped to the right or left to indicate a positive or

negative response. This pattern is diff erent from the swipe views we

talked about in
navigation patterns. Here, the swipe gesture is being used for an action rather than
just browsing.
Some apps combine the two kinds of swipe patterns, for example Carousel, which
lets you browse through multiple photos by sliding them to the side, as well as manipulating them by swiping upwards or downwards to share or hide them. Mailbox
popularized the side-to-side swiping actions for email clients, allowing you to mark
emails as read and schedule them for follow-up by swiping right or left, respectively.
Secret let's you discover new actions the way it let's you discover new menus. Swipe
left on a secret and you like it.



Silvia Weinzette...
Lead Designer at BAB design studio

Jimmy Smith
Founder at Tapview

Dr. David Comfort, D.Phil
RE: Investment Opportunity for ReLife...
Dear Robin,
...

Michael Johnson
RE: App Design/Meeting
I'll be in NY April 22-24, would you like to do lunch Wednesday at 12:15? ...

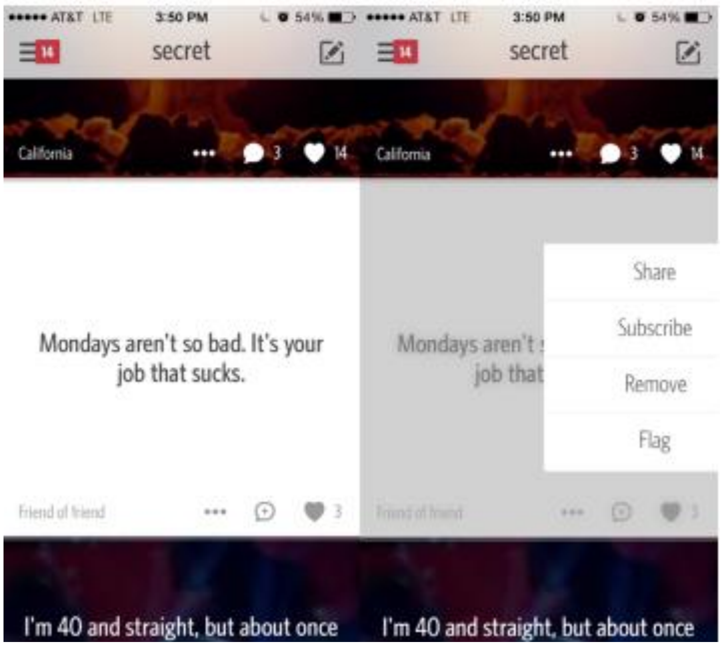
Michael Johnson
App Design/Meeting
Hi Robin,
...

I've stopped add[illegible]hai. The kahwa tastes better.
Caution: the images on box are deceiving. The food inside is nowhere similar to the images

III. Problem
The user wants to know about new activity or actions they should take at a glance.
Solution
Highlight recent activity by visually marking new content.
There are several implementations of this pattern. For example, placing a numbered badge on the label
was popularized by iOS but can be seen in many other apps now such as LinkedIn,

Facebook or Quora. Twitter does this as well for the notifcations button but also has

a small dot on top of the timeline icon to indicate new activity in a more subtle way.
Another way to display notifcations is with a banner that drops down within the
app to show new activity. The Facebook app does this as well, showing a small popup when there are new items in the newsfeed.



Mondays aren't so bad. It's your job that sucks.

I'm 40 and straight, but about once

Problem
The user wants quick access to controls that are secondary or only relevant to
specifc sections or content in the application.
Solution
Clear up the clutter and let users discover particular actions only when they need
them. These invisible controls can be accessed by any gesture - swipe, tap, double-
-tap, long-press etc. (which we talk about in the gestures pattern). This gives you the
ability to keep these actions off -screen, saving some valuable real estate. Secret, for
example, uses gestures instead of visible controls. Swipe right and you’ll

expose an
action menu, which is a minimalistic version of a drawer pattern which we’ve covered earlier. When creating content, users can swipe horizontally or slide their fnger
vertically across the background to change its color and pattern, or in case a picture
is being used, its brightness, saturation or blur. There are no other controls that let
you do this - nor should there be. This UI design pattern is so intuitive and clean
that you’re bound to see a lot more of this type of interaction. Pinterest is another
app that uses gestures to hide action buttons. A long-press on an image reveals
buttons that let users pin or comment on it by dragging the pop-up control to the
button.
Uber is an alternative implementation of this design pattern. Uber also let’s you
toggle between booking a ride and seeing the fare estimation by tapping the slider
button once you’ve chosen which ride type you want. This is a simple yet important
UI design pattern that makes me smile every time I’m doing fve things while trying
to get a ride somewhere, but want to make sure Uber isn’t ripping me off with surge
pricing. Snapchat and Facebook Messenger let you access features when you need
them by swiping any friend left.

IV. LIFE SAFETY

4.1 Ecology

4.2. Safety methods

4.3. Radiation of smartphones

CONCLUSION

We often come across clients with great ideas for a new app, and they wish to have the app developed and got into the market as soon as possible. But the truth is that even when using lean software development approach or Agile methodologies, it still takes time and money to build an app.

While various factors come into play when building a mobile app, it takes a minimum of three months to make a mobile app's initial version, and it takes more than six months to build apps with advanced functionality and more features.

To get a rough idea of how long building a mobile app takes let us take a more thorough look at the process of production from start to finish.

All apps begin with an idea. It is this idea that is usually refined into a firm basis for a mobile application.

The discovery or inception phase is about refining the idea for the app. To build a successful product, it's critical that you ask yourself some fundamental questions. The following are some of the factors you need to consider before you publish your app in any app store.

Competitive Advantage: does the market have similar apps already? if the answer is yes, in what way does your app differentiate from those?

Infrastructure Integration: what are the existing infrastructures that the app will integrate or extend? Also, apps need to be examined in the context of mobile form factor:

- Form/Mobility- how will the app work in a mobile form factor? In what ways can I add value using technologies like camera, location awareness, etc.

- Value: what value will the users get from this app? How are they going to use it?

To help with functionality and design of an app, you may want to define Use Cases and Actors. Actors refer to roles with an app and are users in most cases. Use cases, on the other hand, are intents or actions.

For example, a task tracking app may have two Actors who could be User and Friend. A user could Create and Share a Task with a Friend. In such a situation, creating and sharing a task are two distinctively different use cases which, along with the Actors, will guide you on the screens you have to build, and the kind of logic and business entities you will need to develop.

Once a suitable number of actors and use cases has been captured, it becomes much easier to start the process of designing an app.

At this stage, you already know the kind of an app you want to build. You've put all the nuts and bolts you need. Now is the time to start system development.

Start by assembling your dream team since it knows all the project requirements. Create a workflow design and chart for the project to set boundaries for certain responsibilities. The role of the team, core responsibilities, and functions are spelled out and documented in this stage.

This will save a significant amount of time and project costs as the development proceeds. It is also during the development stage that personnel are trained with regards to any extra requirements of the project.

The rule of thumb is that mobile developers should first make a lightweight prototype to determine whether it is functional and feasible or not. The prototype is also important in seeing if the project has access to required technologies that will achieve the app's aims from the start.

Once approvals on the tasks assigned to relevant departments are obtained, it’s time to go to the next stage.

BIBLIOGRAPHY

Used internet resources

1. https://docs.microsoft.com/en-us/xamarin/cross-platform/get-started/introduction-to-mobile-sdlc
2. https://www.sourcesoftsolutions.com/reasons-that-explain-the-increasing-demand-of-mobile-app-development/
3. https://www.quora.com/Why-do-we-need-programming-language-in-computer-programing
4. https://www.quora.com/Why-do-we-need-different-computer-programming-languages
5. https://developer.android.com/topic/libraries/architecture/
6. https://developer.android.com/topic/libraries/architecture/room
7. https://developer.android.com/topic/libraries/architecture/livedata
8. https://developer.android.com/topic/libraries/architecture/viewmodel
9. http://greenrobot.org/greendao/
10. http://greenrobot.org/eventbus/
11. https://firebase.google.com/docs/storage/
12. https://firebase.google.com/docs/cloud-messaging/
13. https://guides.codepath.com/android/consuming-apis-with-retrofit
14. http://square.github.io/retrofit/
15. https://lvivity.com/5-phases-mobile-app-development-lifecycle

Appendix