Muftau Tunde Abdulrahamon

Development of a mobile workout application

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Development of a mobile workout application

20 November, 2016

Author(s) Title Number of Pages Date	Muftau Tunde Abdulrahamon Development of a mobile workout application 46 pages 20 November, 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Instructor(s)	Kari Aaltonen, Project Manager

The application is aimed at developing a workout model for people who needed to perform exercises in a conformable procedure. GYMKit is a mobile application designed with Android technology. Regular exercise has health and physical benefits which are hard to ignore. Health and physical fitness are the fundamental targets for the Application. Basically, GYMKit is implemented in such a way that the collection of implicit routines and that which is prescribed to the trainees by their trainers are in one component. GYMKit is a form of hub or component for categories of built-in and custom workout routines.

GYMKit is designed to contain the built-in categories of workouts. The collection of workouts is meant to contain descriptions and procedures on how the workouts should be performed. The collections are the ones recommended by experts and each category of workouts has a set of another workout. Users can also customize their workouts, either with personal training plans or routines given to them by their trainers for the purposes of monitoring their physical and wellbeing outcome.

Perhaps, some trainers do monitor their workout tasks by keeping the accounts on a booklet. GYMKit application has a component that stores and monitors user's workout routines. GYMKit offers a great intermediate utility between trainers and trainees. However, it is strictly encouraged to monitor the consequence activities in the application for the impact to be felt.

Keywords	APIs, SDK, iOS, CRUD, DVM, GYMKit, XML, Gradle.

Contents

1. Introduction

People have been avoiding active tasks these days, partly due to the facts that more and more technologies have been invented to make our lives easier and less rigorous. We drive and work in the office, sitting for hours. Machines make our laundry. We entertain ourselves with TV programs for unreasonable hours and a few people engage themselves in physical work. (Emis Group 2016)

We move around less and burn off less energy than people used to. Research suggests that many adults spend more than 5 hours a day sitting down, at work and during their leisure time (Park 2010). Inactivity is described by the Department of Health as a silent killer In United Kingdom. Evidence could be seen that idleness, such as sitting or lying down for longer periods, is terrible for the health of an average human. (UK Department of Health 2012)

People need to get involved in one activity or another to ward off diseases, improve quality of life, control body weight and keep fit at all times. It has been proven that eating a balanced diet alone is not enough for body fitness. Performing a regular exercise for different parts of the body would keep the muscles of the body in healthy condition and good shape. (BodyBuilding 2016)

1.1 Overview

GYMKit is a project designed to support and serve as a guiding tool for trainees and aiding mechanism for trainers both at home and at the GYM. The project is an Android Application targeted at people who like to keeps fit and keep track of their workout tasks and sets of workouts done at regular interval. A huge number of people work out without any workout plans, some do have plans but not well documented.

What would have been the situation if the workout and set of workouts were selected from an application on a mobile device? GYMKit is one of many Workout applications developed for this purpose.

GYMKit aims to design a better workout application with more features that make it stand isolated among others. Users are able to customize their collection of workouts to their own taste and keep abreast with tasks performed from time to time. This experience could only be served from a mobile gadget and not having to hinder the users from using other apps on their mobile devices.

1.2. Objectives

GYMKit provides some important features, which are important to people while working out both at home and in the gym. Most exercises people like to do in the gym are all contained within the application. People perform cardio workouts, biceps, triceps, and abdominal, back, neck etc. With these, users can select any of the available workouts within the above mentioned categories.

Users also have the opportunity of customizing their workout categories by adding their needed workout or that given to them by their instructors as part of monitoring strategy. There is a task implementation for every workout to users and after every workout, the user's activity would be stored in the database for future feedback or estimation when needed by training instructors.

Part of the customization of the workouts is the notification of users for workout activity. Users can enable the notification features, which alarms them of upcoming exercises to be performed. Another customized feature is to add category of workouts to favourite menu for quick access.

Some of the important components of the application would be discussed in the consequent chapters. The application architecture alongside with its implementation will be touched upon. GYMKit application testing, which is an important concept in the development of the project will be discussed in chapter 6.

2 Mobile Application Development

Mobile application development is similar in one way or another to Web application development and has its fundamental purpose as being the creation of applications that could be built, executed and used on mobile platforms. However, mobile applications are often written specifically to take advantage of the unique features a specific device comes with in terms of both hardware and platform utilities.

Mobile application performance is always at stake when it comes to usability. Users appreciate the richness of an application. Handheld devices are designed for this purpose. All the rich components of the architecture that are expected to be incorporated are designed within the system such as the execution speed, specified processor, virtual machine and a whole bunch of APIs. The end-result of this is that the applications would be written for a specific processor of a device and an operating system. If any mobile application needs to run on similar operation system, the same application only need to be installed but perhaps, should be rewritten for a different operation system.

Statistically, most mobile applications use both Android and iOS (IPhone operating system). These two technology has been used in so many mobile technologies and most often, each technology has modelled the operating system to suit its needs by using various configuration techniques that comes with the operating system. Every operating system is orchestrated to work with the applications designed for itself. Android application cannot run on the Windows platform and vice versa. (Dogtiev 2015)

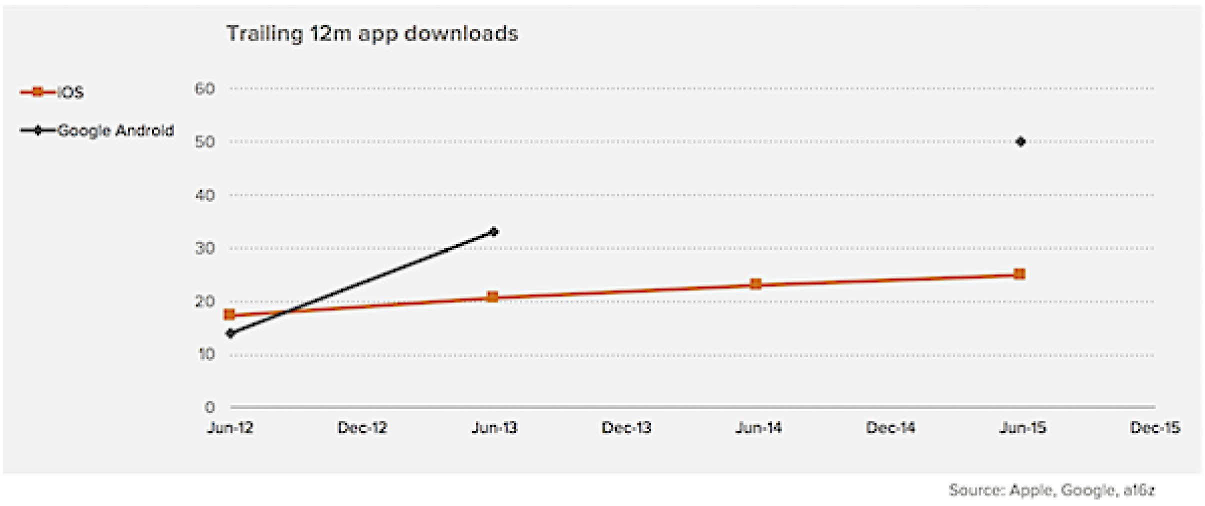


Figure 1: Usage of the iOS and Android technologies (Dogtiev, 2015)

Base on figure 1, the Android platform has continuously outgrown iOS in terms of market share of downloaded applications. In the financial year quarter conference, which is a worldwide developers conference, Apple had announced the number of iOS apps that were downloaded. The graph demonstrates android and iOS apps ecosystem and market growth. However, the popularity of both platforms have increases robustly and it is now, no technology has not been able to slow down their pace.

2.1 Native Application

Native applications are Mobile applications that run locally on a mobile device. These apps are downloaded from the platform's store.

Apple applications are downloaded from the App store and Google apps are downloaded from the Google store. The stores, which are usually cloud storage, store applications created by developers and they could be published for sale or for free. Android application has, as it claims to be, had a huge portion of market share of free application deployment as shown in figure 2.

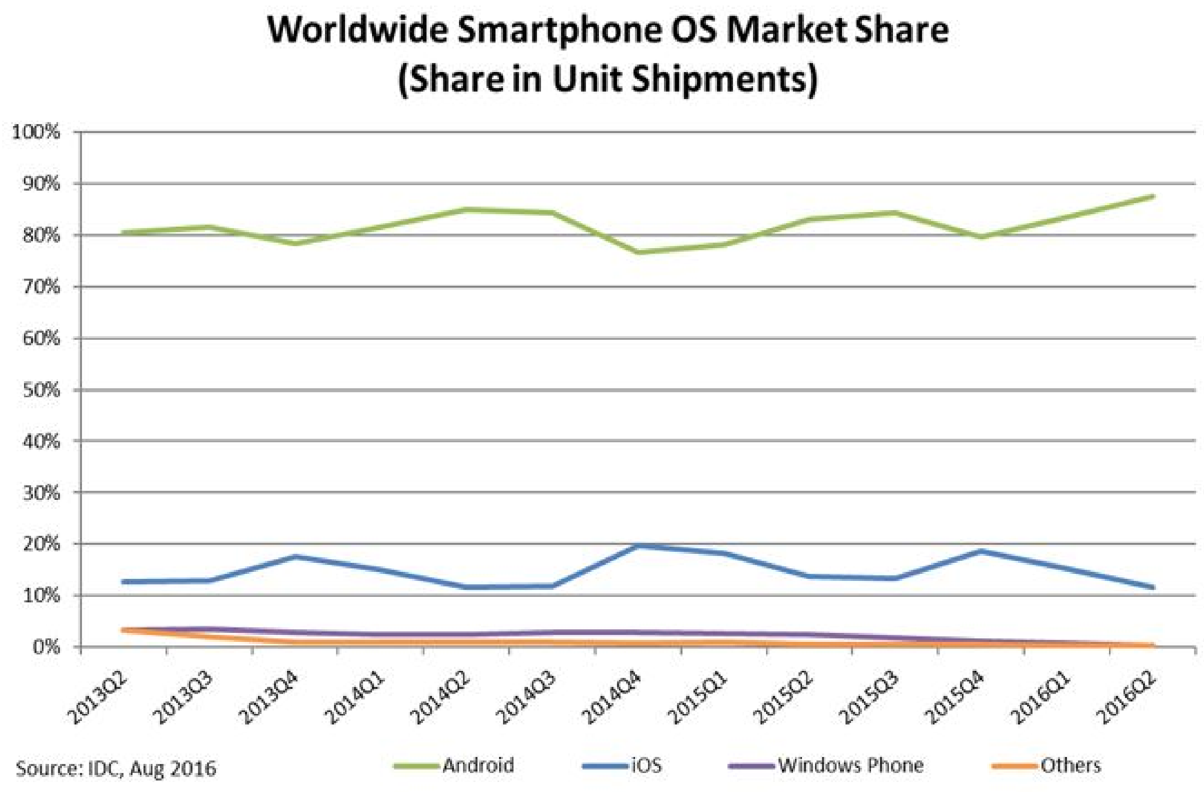


Figure 2: The Market spread of Android Technology. **(Idc 2016)**

What distinguishes native apps from their competitive alternatives is that they are designed for a specific kind of device. For instance, iPhone applications are written in Objective-C, Android apps in Java and BlackBerry with Java.

Each mobile platform comes with their own development tools to give developers what to work with, user interface provisions, virtual machine and Android SDK. With these developers are empowered to develop a native app at ease.

There are a few advantages to writing applications natively:

- They give the utmost speed, reliability and responsive features to users.
- They can make use of more functionality of the devices including the camera, recording, compass, Global Positioning System (GPS) and swipe gestures.
- Publishers can make use of push-notifications, notifying users every time when there is new content published.

This is not only about the access or control over the device itself, but the platform offers developers of native apps fundamental layout capabilities. Developers are free to make use of the features, which gives the design a more responsive approach than just a web app.

One of the major disadvantages of native apps is that they will not work with other kind of operating system. When developing a mobile app in Objective-C for iOS, it is not going to run on Android without being entirely re-written in Java. When building for multiple platforms, developing a native app can be quite expensive.

Native apps are offering the best user experience. When building from scratch, developer should be aware of the cost ineffectiveness and support.

Considering the cost of app development, developers or agencies could charge up to €15,000 to €40,000 on a native app built from the scratch (Yarmosh 2015). With native apps, some features or support don not have to be built again. What is needed is that the support components would be indicated on the current application as if the component was part of the application. Platforms or application designers mostly design for a specific set of functionalities. The figure 3 below shows few of the mobile applications operating systems

Figure 3. Platforms capable of creating native Applications. **(Continued Learning 2015)**

2.2 Web Application

A web application is a mobile version of native apps. Web applications are displayed with a web browser, like Safari or Google Chrome. The web application appears on the browser like every other web application. The audience does not have to install a web app. There is no need for more available space in the device in question.

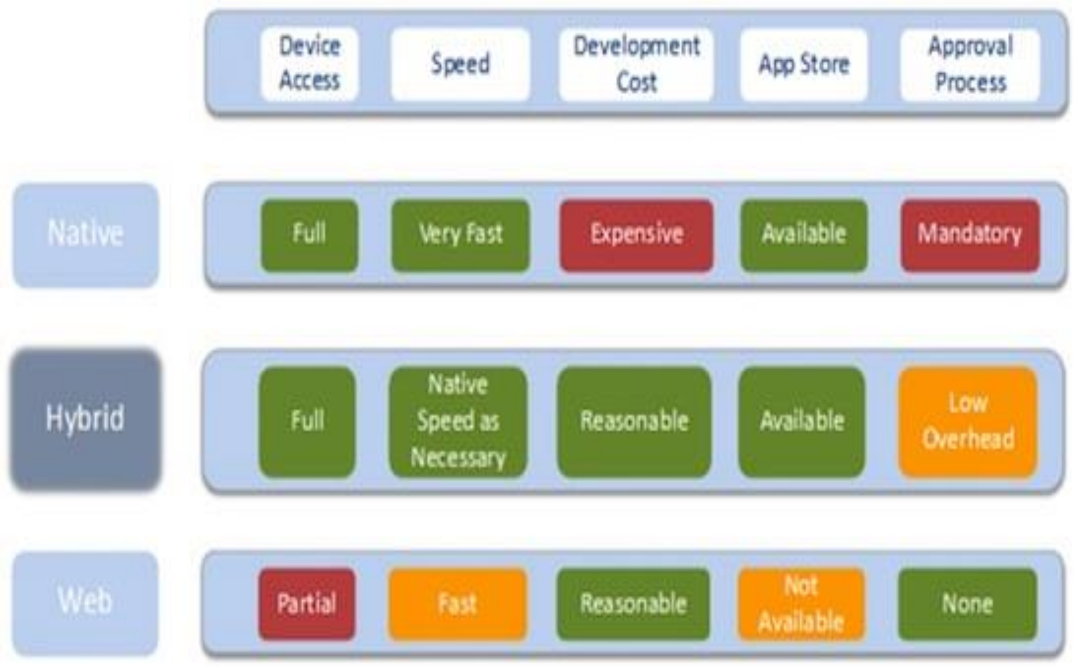
Web applications are designed to look and behave like apps and are commonly ideal when the purpose is simply to make content or key routines available on mobile. Web apps use JavaScript, modular JavaScript, CSS, SCSS, HTML5 and other JavaScript frameworks. A developer will not granted any permission to the underlying SDK provid-ed in Android architecture. Developing a web app is simple with template-like struc-tures.

There are web application experts out there that are convinced that web apps are equal to and came to replace the native apps. Their arguments are restricted to cost and functionality effectiveness. (Borodescu 2013)

Web apps do not depend on any on any of the underlying native apps to make use of any of their ready-made components or features. More implementations still need to be included in the web app architecture for it to be fully called to attention for full replace-ment of native apps. However, proponents of web apps will not agree with this stand.

Web applications have their own limitations mostly in terms of what they could perform efficiently. What web applications require always is the connection to the internet. (Kari 2016)

It is not possible to send notifications and make them appear in the app content. It is difficult to use from the user standpoint. In addition, when using web app, because it relies on Internet connection several features might be lost during transmission and would be faced with security risk as well. Thus, native and hybrid apps appear on the App Store and Google Play. With millions of searches every day on these stores, there is always a need to give the users what they would appreciate. (Matt 2011)



	Device Access	Speed	Development Cost	App Store	Approval Process
Native	Full	Very Fast	Expensive	Available	Mandatory
Hybrid	Full	Native Speed as Necessary	Reasonable	Available	Low Overhead
Web	Partial	Fast	Reasonable	Not Available	None

Figure 4: Some features attributed to the three forms of mobile applications. **(Dar 2015)**

2.3 Application Architecture

Mobile application architecture is essentially the implementation of tools and procedures to construct a mobile application with the design models being considered. The design and techniques used will be compatible with every platform that supports the desired architecture and figure 4 shows some of the features attributed to the mobile applications development. The architecture in context in most cases is understood to be the platform on which every mobile application would be developed. The Android architecture is actually based on the android platform and every application that is executed on it uses the underlying structure of the platform. The same scenario is for the iOS architecture. The underlying architecture basically lead developers to consider some certain challenges that have to be acknowledged while developing any kind of mobile application. (Dar 2015)

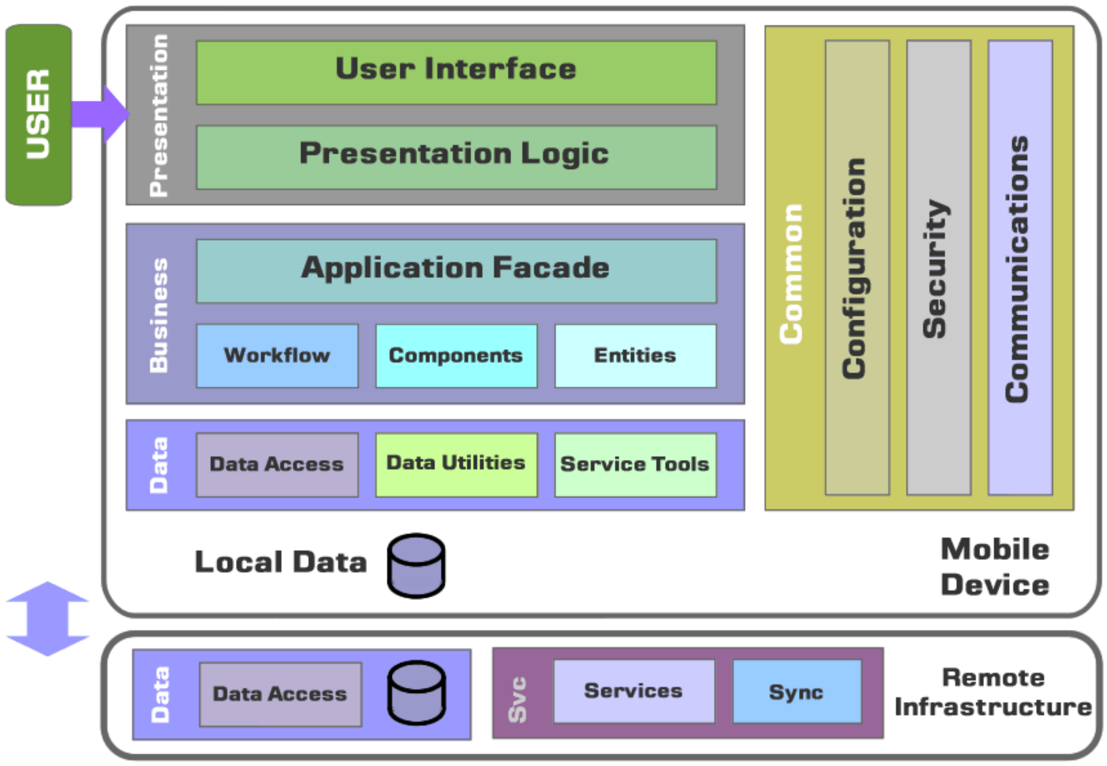


Figure 5. The mobile application architecture. **(Kari 2016)**

2.3.1 Mobile Application

A mobile application is a software application developed basically for hand-held gadgets such as smartphones and tablets that are structurally different in terms of architectural layout to their contemporary desktop devices. Mobile applications market has grown so much that a huge number of application have been developed for different platforms and these developments have never stopped to create jobs in the technology market. (Kari 2016)

Mobile applications are designed based on the yearning of the people for the need for new technologies in the market. These applications can be pre-installed on the device at the time of production or perhaps installed by users when there is a need for more applications to power their needs. The developers of these applications also have to consider certain aspects such as the screen sizes of the device in which the application would be executed. The configuration of the various platforms, hardware specifications, platform updates, changes in the APIs and the best practices. (Al Salool 2012)

A mobile user interface is one of the most important features of all mobile apps. It is actually the presentation logic of the apps and the user tends to appreciate the application even before using it by looking at the user interface. Developers would have to be forced to conform to some features such as the layout, application widgets, compound layouts and the design, and in most cases would have to rely on some third-party libraries for this purpose. The user interface is the view component that exhibits the hardware and the logic of the whole application. (Continued Learning 2015)

In the next section I will discuss some of the key design provisions that developers should consider when developing a mobile application. They are the fundamentals performance proficiencies of mobile applications.

2.3.2 Device Configuration Management

Configuration management is a technique used in describing the software and procedures that are necessary for the secure distribution of mobile applications on mobile devices. The configuration offers a reasonable solution such as software licensing, authorization and authentication, application lifecycle and application limitations.

The underlying platforms also provide management features. Failure of the application to conform to this configuration exposes the application to various attacks that obviously result from administrative mal-implementation from within the mobile application.

When designing an application with configuration management in mind, developers need to consider some requirements such as configuration stores which have sensitive data within it encrypted, implementation of privileges and allowing authentication for access permission to restricted configuration files. (Microsoft 2016)

2.3.3 Authentication

Authentication must be taken into consideration when developing a mobile application. It is the basis on which application security relies on. Hijackers and web crawlers always look for loopholes in mobile application connections and browsing. Thus a good authentication implementation creates a reasonable amount of security against unwanted privileges. (Degges 2015)

A strategy needs to be developed to identify those that have access to the datastore. Only authenticated users have access to the right data stores. Soring the password in a plain text is a bad principle when it comes to mobile application design.

Storage of password in a plain text must be avoided in the database or whatsoever. Most mobile applications are designed to have an internet access which means data request and response are probably over the internet. Users data are conveyed from the device to the database in a remote location. This information is not expected to get into the hands of unwanted users. That is why important user data such as passwords should be encrypted during transmission for proper accountability and credibility of user data. (Microsoft 2016)

2.4 Application Technology

The rate of technological development and successfulness in the mobile market has given rise to so many challenges that prompt developers to strategize plans for major projects to set the ball rolling in the mobile development proliferation. However, since the beginning of the smartphone revolution, the iPhone IOS development encompasses the mobile market, just recently Google Android has overtaken iPhone in terms of market share. The Google targeted strategies and the platform’s nature which is an open source architecture. Other mobile operating systems that are part of the mobile market share are listed below. (Al Salool 2012)

- BlackBerry.
- Windows 10 Mobile.
- PhoneGap.

When considering how to incorporate mobile technology into existing business logic, the primary issue for both clients and developers is currently the choice between native apps and web applications - or a combination of both. The advancement in mobile technologies has lead to the development of businesses in other sectors that are necessary to the mobile technology such as the hardware sector and application programming interface (API).

Native platforms (Android/iOS) or Hybrid (Cordova/Phonegap) are important platforms to remember when considering a mobile strategy. Developers are faced with many challenges when considering the users of their apps, whether to focus the design on a Web application or on native apps. Both technologies are solutions to the design issue, due of the number of platforms operating, and the best path for the development logic may be different for different platforms.

There is also a huge competition in mobile software, which has brought about a robust change within each platform. Both Apple iOS and Android have undergone a powerful change with update and release, with the intention of supporting users with extensive features. Few platforms such as Android for instance – are being deployed on devices produced by different manufacturers. Each of the mobile developer always wants to make changes to the operating system configuration. There is a need for them to consider some additional features that would make the platform architecture somehow different from another, yet with the same operating system. Nonetheless, Android hardware is now available in the mobile market.

At the early stage of smartphones, consumers that brought the devices were rich people. But now smartphones are now available at a lower cost, and the statistic shows the total number of mobile phone users in Finland from 2011 to 2019. Growth in the number of users is expected to remain slightly unchanged: from 4.5 million in 2013 to 4.6 million by 2017 (Idc 2016). As a result of different releases and manufacturers, developers have to consider multiple screen sizes, hardware specifications, API updates and configurations.

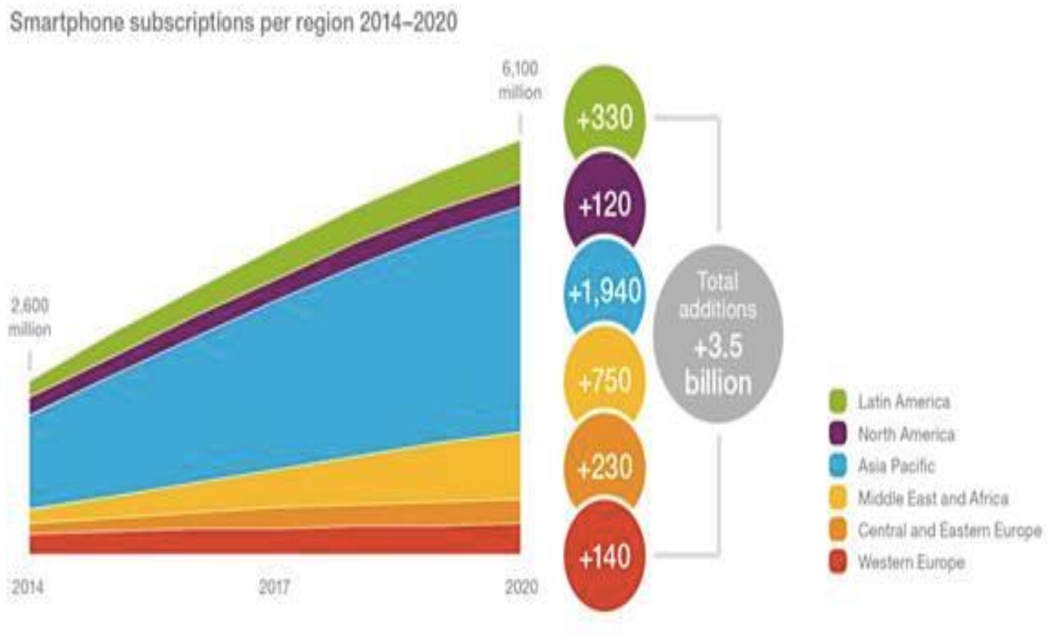


Figure 6. Number of smartphone users around the world. **(Idc 2016)**

3. Android Architecture

3.1 Android Application Framework

The Android platform is an operating system that is a stack of software components that are categorized into five layers. The following is the operating system component stacks. (Al Salool 2012)

- Applications
- Application framework
- Libraries
- Android runtime
- Linux kernel. (Al Salool 2012)

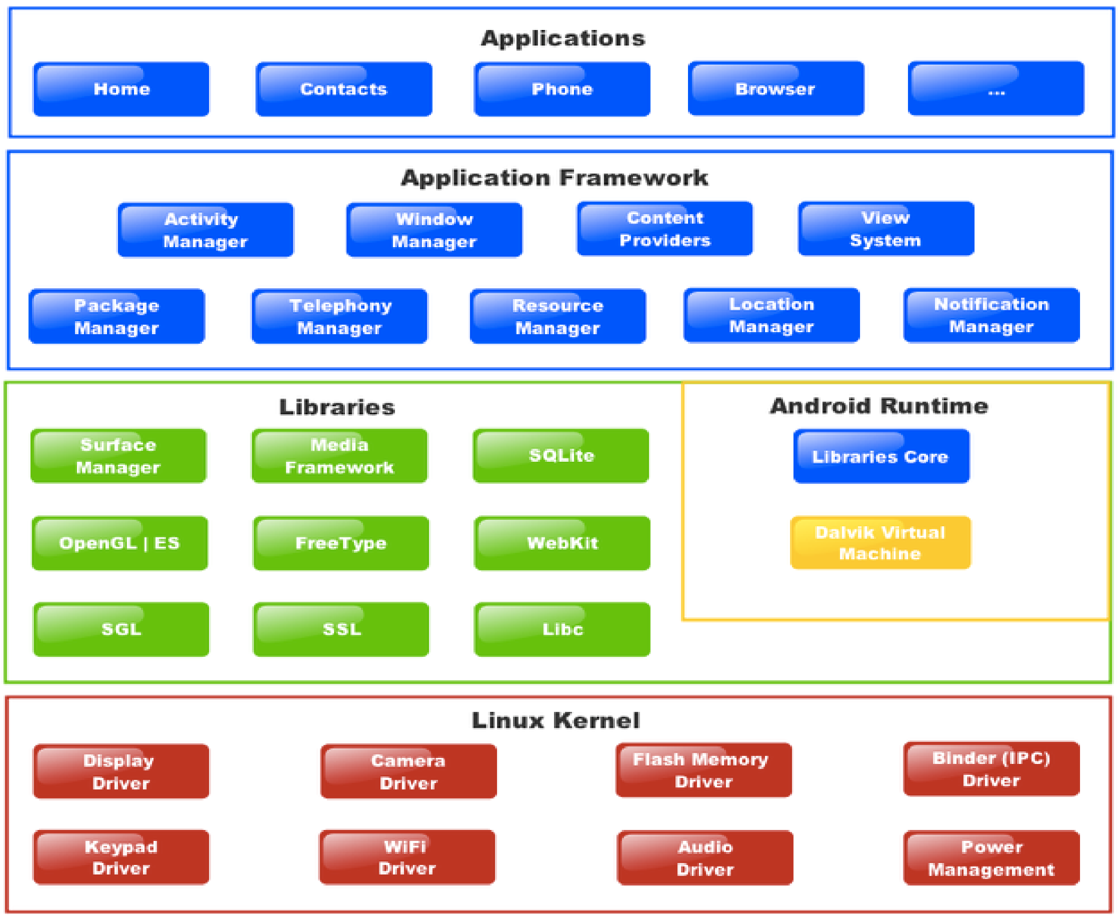


Figure 7. The Android Platform Architecture components. (Al Salool 2012)

Applications

Android application is at the top of the components stack. Android applications are developed and installed into this layer. This layer is the one that is fully under the control of the application users. Some of the examples of the applications are games, dictionaries, File Viewers, banking system applications.

Application Framework

This is the layer that provides some services to the underlying applications. These are higher-level services made available to the applications. The services are modelled in the form of Java classes that applications users are allowed to make use of.

Some of the Android framework services are listed below.

- **Activity Manager –** The application lifecycle and the activity stack are control by the Activity Manager. The package of services provides a bunch of services needed by developers that would be used in their apps.

- **View System –** The view system is the building block of the user interface components. Android provides services on this basis in the form of widgets for an interactive UI features on applications. user interface is by far one of the most important components of Android design.

- **Resource Manager –** The manager provides access to resources that are embedded. These resources are non-coded components that are used in the user’s application Java code. The resources are referenced in the applications.

- **Notification Manager –** The manager allows applications to show alert messages and contains management features for notification settings.

- **Content providers –** Applications require some other features that are needed by other applications. The data can be shared among the applications. (Android 2016)

Android Runtime

The android runtime system is an important layer of the architecture and contains the Dalvik Virtual Machine (DVM). DVM is a Java virtual machine specifically orchestrated and used for Android platform.

The DVM uses the Linux kernel components such as the memory management and the multi-threading features, which are vital to the java language. The VM provides the environment for the execution of each Android application in its own process and attributed with an instance of the Dalvik virtual machine. The runtime also has the component core libraries used by developers in the form of standard java programming language for writing Android applications.

Android Libraries

This section is composed of Java-based libraries mainly written for Android development. The major library in this category includes the Application framework.

3.2 Development Tools

Initially the Eclipse ADT is the Android Development tool (IDE), which is a plugin that is no longer supported by Android platform. The official IDE for Android is now Android Studio.

Most Android projects are now developed with Android Studio. Gradle is now the supported method for building Android Application. The GYMKit project is entirely developed with the Android Studio tool. There is no need for any migration of the project. The Android studio consists of a major features that are required for the development of Android applications with ease.

3.3 Platform version and SDK

The latest release of update for the Android distribution numbers for each Android version has been made known by Google. Android Lollipop continues to be the most favoured Android version. The number of users continues to increase. GYMKit Application is developed with Android Lollipop version with the required API level, which is 21 and 22.

Android SDK is a development toolkit that contains all the needed libraries for the development of Android applications. It also composes of working apps with codes and the Android virtual device manager. With this, users can test applications on any version of Android on devices that runs Android platform. Different components including the platform tools, Android Debugging Bridge (ADB) and the build-tools can be downloaded.
(Android 2016)

3.4 Android Virtual Machine

Every Android Application runs in its own process with an instance of a Dalvik virtual machine. The Virtual machine is designed in a way that multiple instance can be executed on a device efficiently without any conflict. The Dalvik virtual machine executes a specific file extension, which is a.dex executable format. The Java language compilers compile java classes into this format.

The source code format of Java language in Android is in x.apk and when built would be converted into .class and finally converted into x.dex executable, this is shown in the figure below. (Scriptol 2015)

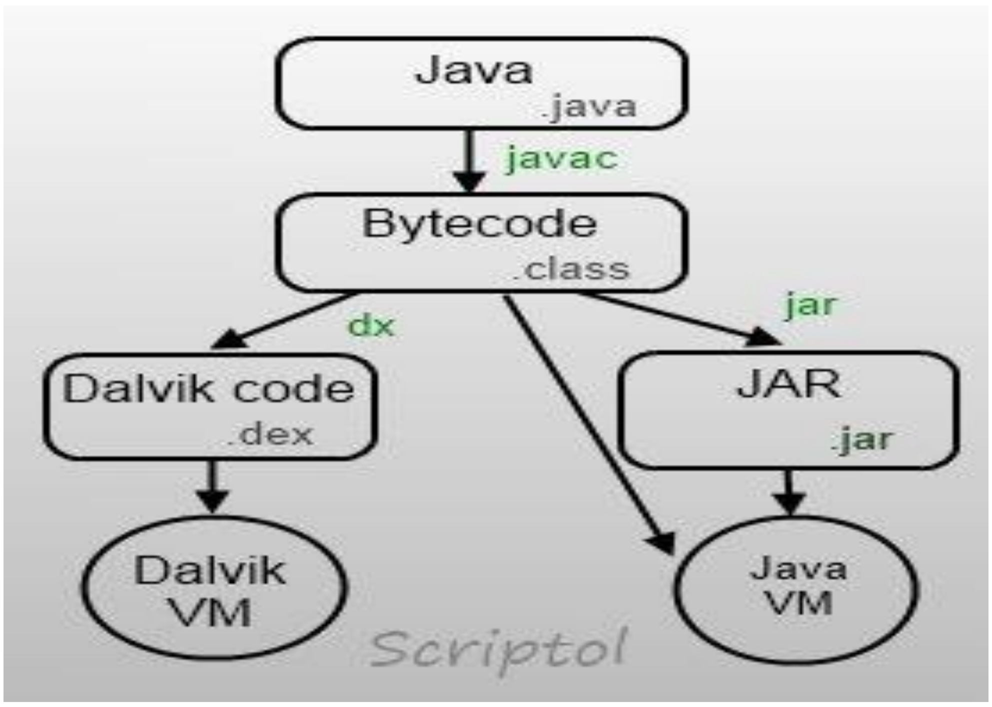


Figure 6: The conversion stages from Java source code the final stage of a.dex executable by the Dalvik virtual machine. **(Scriptol 2015)**

3.5 Networking Features

The Android technology is a huge platform that offers a number of user interface components and possesses the potentials of handling data in a more robust manner. The ways in which it implements Networking functionalities add more to its enriching potentials. Most applications are developed with the basis of connecting to the outside world, be it fetching data from the server or playing an online game. The Android architecture provides a robust set of features which allows developers to connect application users to the internet for one or more reasons. (Al Salool 2012)

Connecting embedded devices or mobile devices to the Internet is made possible through the networking packages implemented into the Android SDK. To perform any Network operation with an android application, the application needs to add the below permission to the application's manifest file. (Al Salool 2012)

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Most of the cases whereby the Android Applications needed to connect to the network, the HTTP method is used to send and receive data from the data-store. When developers download the Android SDK, the developer now has the so call HttpURLConnection client, which is a package for making the connection through the user devices. The package supports the following features.

- HTTPS
- Streaming uploads and downloads
- Configurable timeouts
- IPv6
- Connection pooling. (Microsoft 2016)

With the networking capabilities, Android applications developer would have the full control of the following.

- The Developer would be able to choose an HTTP client
- Developers would be able to check the network and manage the connection
- Performing Network Operation on a separate Thread with ease

- Connecting and Downloading Data from the data-store
- Finally converting the obtained InputStream data to a string for Application usability or view rendering. (Microsoft 2016)

Designing User Interface

User interfaces in Android platform are built using the Android layouts that may be different for all types of devices. With the flexible framework provided by android, the developer can be able to offer their users different sorts of UI designs that allow on Android application to display different layouts, modelling the widget with custom ones.
The Android framework offers the following user interface potentials:

- Developers would be able to develop Applications for multiple screens. The layout would be flexible enough to fit perfectly on any screen size and the interaction concepts are designed in an adaptive manner.
- The developers would have the capability of adding different sorts of widgets to the layout using the support libraries. For instance the toolbar support library could be used to implement application bar on any varieties of devices with so many properties developers can render to the views.
- The user interface features also provide the developers with features that allow the showing of pop-up messages to the app users, also with the support library such as Snackbar widget.
- The Android Framework also provides the developers with ability to customize the layout widgets to more suitable and interactive views
- Being able to maintain compatibility is a vital concept in API development. With android UI components, there is a possibility for the recent versions of the platform to be compatible with the older ones due to the backward-compatibility of the UI APIs included in the SDK.
- Implementation of accessibility features into the Android SDK makes it more sophisticated for creating a UI for physically disabled people.
- Developers would be able to create a UI with rich APIs such as those used in implementing the material design. This entails the definition of custom animation, management of drawables, inclusion of material themes and main-

tains the compatibility of the material design with previous Android UI designs. (Rouse 2016)

Concept of Usability

Usability can be defined as the ease of use of a mobile device or an application. In this concept, I will be referring to the ease of use of a mobile application. The aim of the use of the mobile application is to get some features and functionality and the application would be difficult to use without the usability being considered. Every application is expected to be effective, sophisticated, and satisfactory and the color and contrast should be intact and follow some other principles that are considered the standard to be followed by developers.

The design of the application should be done in such a way that users of all abilities would be able to use the UI efficiently. Also those with different disabilities such as hearing impairment, low vision, or blindness should be able to engage themselves in using the apps.

Users of all apps should be able to appreciate the color and contrast of the mobile applications. Developers should also take into consideration the sound implementation of the app, which is an alternative to the visual implementation. Unnecessary sounds should always be avoided and the sounds that interpret screen elements or content should be designed for a correct or almost correct efficiency.

The design of the applicaton should also respond to user input effectively with the touch target performing at its maximum functionality. The touch targets are the areas that respond to the user input. The touch target should be at least 48 * 48 dp to ensure balanced density and usability of UI widgets and information. The recommended size of touchscreen widgets is 7-10 mm. (Dar 2015)

Visibility and Accessibility of text content are also primary concepts in this context. The accessibility text is read by the screen-reader Software such as Android Talkback. The Accessible text includes visible text and nonvisible content of the Applications. Both visible and nonvisible text should be descriptive enough. Developers should test the app with a screen-reader before launching the app to the public. (Al Salool 2012)

With the use of the standard platform controls, applications contain the necessary mark-up and the implementation visual contents. Every platform has its own assistive technology. The app should be able to adapt to this to give users the efficient experience that they suit. (Dar 2015)

Scope of Human Computer Interaction

The human computer interaction concept lies within the scope of the design and the usage of computer technology taking into consideration the interfaces between the people and the computers. People relates with the system in a certain way which is considered as a standard to developers of applications since developers develops applications to satisfy their respective users. Paying so much attention to the design with the users in mind is the hearth of application framework. So many applications users still believe that enough attention has not given to user oriented design procedures. The ease of use should always go along side with the design of any kind of computer application. The study of the user preferences should be considered at the inception of the development phase due for it changes from time to time. (Rouse 2016)

4 GYMKit Back-End Architecture

4.1 Parse Platform Overview

The Parse technology is a BaaS platform simply stand for Backend as a service. The platform provides developers with a range of development software development kit (SDK) for the creation of different kinds of applications on various technologies or platforms. Android, iOS, Windows and JavaScript apps can be developed using their respective Parse SDK. Each of the technology SDK contains various classes and methods needed to incorporate backend features to applications.

The technology has the following main features:

- Parse Core: This is a feature in Parse that is used mainly in saving applications data to the backend Database.
- Parse Push: This gives notification features to applications.

GYMKit application uses the Android SDK for its backend. The cloud server, push notification and analytics components are the features of the platform utilized. The rest of this chapter focuses on detailing the implementation of the parse platform for the development of the GYMKit applications. The Parse platform offers as backup option that is in the form of JSON file. The backup could be imported back to the parse data browser if need be. Parse server developers now have the benefits of including index managements, performance tuning and backup and restore functionalities. (Parse 2016)

4.2 Android SDK

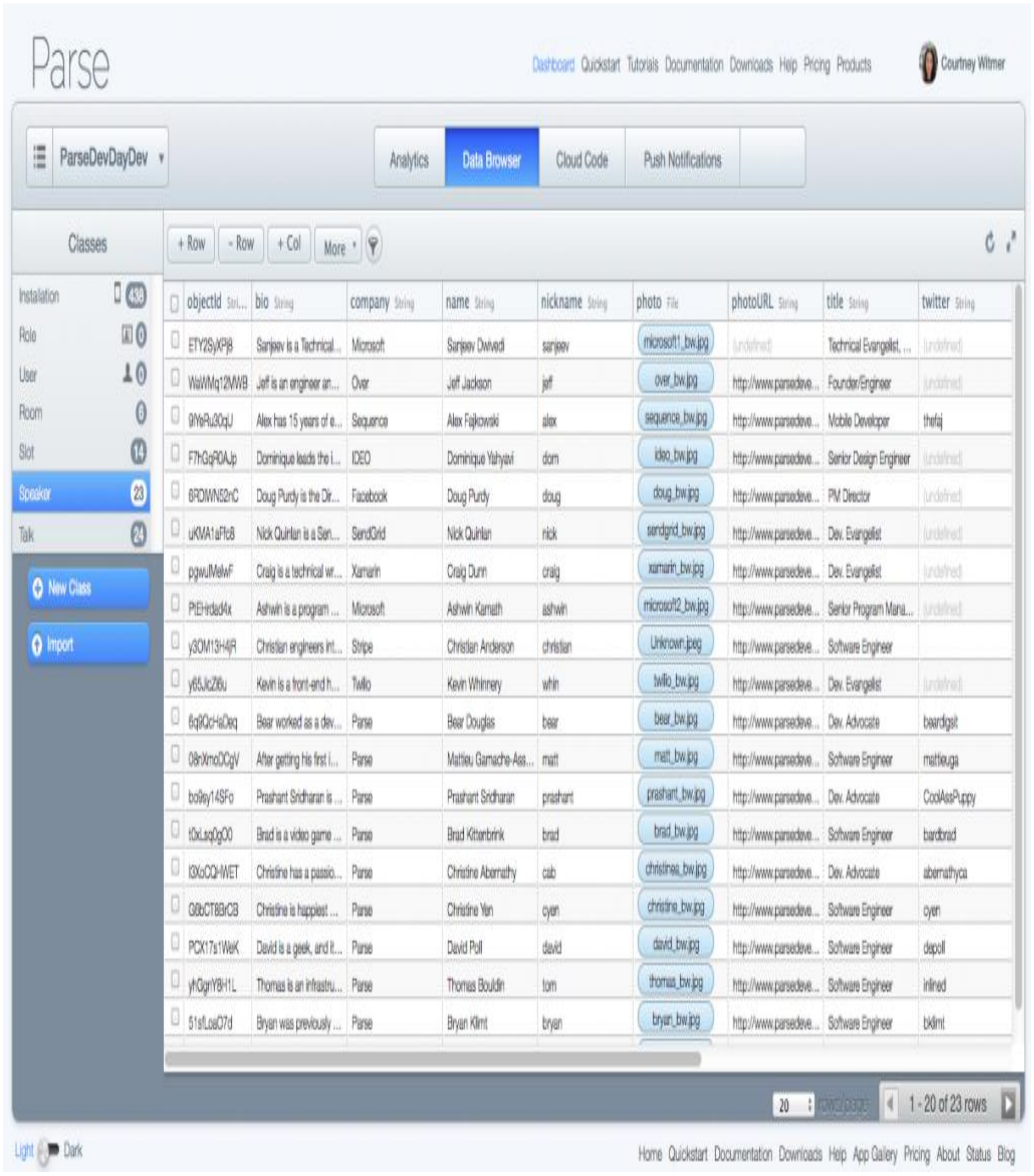
Parse provides a variety of SDKs for mobile app development. The SDK contains enough APIs that contain several classes for the development of applications with ease. GYMKit uses Android SDK tool for its backend environment.
The following are some of the objects that are contained in Android SDK:

- The ParseObject: This object is tasked with the all functionalities that pertain to storing data on the parse platform. This object contains key-value pairs that are compatible with JSON data.

- ParseQuery: The Android SDK provides the query object for the purpose of fetching multiple data from the Parse and also putting conditions on the objects retrieved from the database. With the ParseQuery, a retrieved object could be cached automatically avoiding the writing of the code every now and then necessary. The idea behind the query is to create a query object, apply a condition on it and finally fetch the matching ParseObject from the Parse using the findInBackground method with a FindCallback. These are a few of the methods provided by the ParseQuery Object.
- ParseUser: In most apps there is always the idea of creating a user account using an authentication means or the other. Parse provides the ParseUser object that is a specialized user class. It automatically handles a user operation that pertains to account authentication.
- ParseFile: This object lets the developer store large files in the cloud. These files would be too large to fit into the store using the ParseObject. ParseFile is used for storing Images, videos, music and any other binary data that are up to 10 megabytes. (Parse 2016)

4.3 Parse Data Browser

The data browser is provided by the parse platform for viewing user data in the cloud. It contains several features that make mobile applications more creative. In the data browser, developers could see their application that was connected to their respective app. Any data that are required to be retrieved from the backend are displayed on the data browser. In the next section I will discuss all the classes used in the GYMKit applications. The figure below shows the data browser of the Parse platform displaying list of classes and objects.



objectId String	bio String	company String	name String	nickname String	photo File	photoURL String	title String	twitter String
ETY2SyXPjB	Sanjeev is a Technical...	Microsoft	Sanjeev Dwivedi	sanjeev	microsoft1_bw.jpg	(undefined)	Technical Evangelist, ...	(undefined)
WaWMq12MWB	Jeff is an engineer an...	Over	Jeff Jackson	jeff	over_bw.jpg	http://www.parsedeve...	Founder/Engineer	(undefined)
9lYeRu30qU	Alex has 15 years of e...	Sequence	Alex Fajkowski	alex	sequence_bw.jpg	http://www.parsedeve...	Mobile Developer	thefaj
F7hGqR0AJp	Dominique leads the i...	IDEO	Dominique Yahyavi	dom	ideo_bw.jpg	http://www.parsedeve...	Senior Design Engineer	(undefined)
6RDWN52nC	Doug Purdy is the Dir...	Facebook	Doug Purdy	doug	doug_bw.jpg	http://www.parsedeve...	PM Director	(undefined)
uKMA1aFfc8	Nick Quinlan is a Sen...	SendGrid	Nick Quinlan	nick	sendgrid_bw.jpg	http://www.parsedeve...	Dev. Evangelist	(undefined)
pgwulMelwF	Craig is a technical wr...	Xamarin	Craig Dunn	craig	xamarin_bw.jpg	http://www.parsedeve...	Dev. Evangelist	(undefined)
PtEHrdad4x	Ashwin is a program ...	Microsoft	Ashwin Kamath	ashwin	microsoft2_bw.jpg	http://www.parsedeve...	Senior Program Mana...	(undefined)
y3OM13H4jR	Christian engineers int...	Stripe	Christian Anderson	christian	Unknown.jpeg	http://www.parsedeve...	Software Engineer	
y65JcZ6u	Kevin is a front-end h...	Twilio	Kevin Whinnery	whin	twilio_bw.jpg	http://www.parsedeve...	Dev. Evangelist	(undefined)
6q9QcHsDeq	Bear worked as a dev...	Parse	Bear Douglas	bear	bear_bw.jpg	http://www.parsedeve...	Dev. Advocate	beardigsit
08nXmoDCgV	After getting his first i...	Parse	Mattieu Gamache-Ass...	matt	matt_bw.jpg	http://www.parsedeve...	Software Engineer	mattieuga
bo9sy14SFo	Prashant Sridharan is ...	Parse	Prashant Sridharan	prashant	prashant_bw.jpg	http://www.parsedeve...	Dev. Advocate	CoolAsPuppy
t0xLsq0gO0	Brad is a video game ...	Parse	Brad Kittenbrink	brad	brad_bw.jpg	http://www.parsedeve...	Software Engineer	bardbrad
l3XoCQHWET	Christine has a passio...	Parse	Christine Abernathy	cab	christinea_bw.jpg	http://www.parsedeve...	Dev. Advocate	abernathyca
G0bCT8BrC8	Christine is happiest ...	Parse	Christine Yen	cyen	christine_bw.jpg	http://www.parsedeve...	Software Engineer	cyen
PCX17s1WeK	David is a geek, and it...	Parse	David Poll	david	david_bw.jpg	http://www.parsedeve...	Software Engineer	depoll
yhGgnY8H1L	Thomas is an infrastru...	Parse	Thomas Bouldin	tom	thomas_bw.jpg	http://www.parsedeve...	Software Engineer	inlined
51sfLoaO7d	Bryan was previously ...	Parse	Bryan Klimt	bryan	bryan_bw.jpg	http://www.parsedeve...	Software Engineer	bklimt

Figure 7: Data browser of the parse platform. **(Parse 2016)**

4.4 GYMKit Classes

The GYMKit classes are the representation of all the workout categories in Parse platform. Workout category will contain some features that will be represented with the columns of each class. The GYMKit app has 13 built-in classes and users of the GYMKit application can create their own custom workout categories, which would then be displayed on the data browser. The application also has some other classes that are meant for other functions. The below list are the classes contained in the GYMKit app on the Parse platform:

- AppUser
- Category workouts
- Biceps
- Cardio
- Chest
- Traps
- Shoulders
- Triceps
- Calves
- Triceps
- Neck
- Back
- Glutes
- Adductors
- Abdominals
- Abductors.

5.GYMKit Application Implementation

5.1 Overview

The GYMKit application implementation entails all the features of the project and its functionalities. The GYMKit implementation includes the special components of Android technology and how the components where combined with the layout resources alongside with the backend components which were discussed in chapter 4. The backend technology plays a robust role in the performances of this project and its functionalities are felt in all the activities used in this project. To begin with, I will analyse the component a little bit in the Environmental setup, explain some key concepts of the development tools and then describe the components of the application itself.

5.2 Environmental Setup and Development Tools

The GYMKit project was implemented using the Android technology and its development tool called Android Studio. The latest version of the Android Studio is being was used to develop this application.

Manifest File: In the form of AndroidManifest.xml is the file that describes the application fundamental components. All the created Activities, Broadcast Receivers, Services and content providers are declared here. The minimum SDK version that is used is 14 and the target SDK is 23. The numbers indicate the API Level, which indicates the supports for different devices and different Android features. That was the API Level used as at the time the application was developed and has the support for some of the current Android newly developed technologies such as the Material Designs.

Build.Gradle: The compilation and the building of a Android application done by the Build.gradle provided in the Android Studio. There is Gradle for the entire GYMKit project and that for each of the projects modules.

Running GYMKit on Real Device: The following steps were taken when setting up my Android device for testing the GYMKit application on real Device.

1. Connecting the Android device to the System with a USB cable. The Development device used for the project is the Android Studio that was installed on a Window Device, so I need to install the USB driver.

2. Enabling USB debugging on device from the settings.

The Android Studio installs the application on the connected device and starts the Application. The application components are essential parts in developing an Android application. The Android technology is configured on the Application level using the Manifest file that is an XML file AndroidManifest.xml. This file describes the interaction of all application components. GYMKit project made use of the components provided by Android. The following is the list and short analysis of the component tasks.

Activity: The activity is the component that every user of an application interacts with. It has a user interface that displays what the users see on the screen. All the lists of the workouts and categories would be displayed with Activity in this project.

For my application to make use of these properties, the activities I used have to extend or be a subclass of the Android Activity class in this way.

```
public class MainActivity extends Activity {  }
```

Services: This component runs in the background to perform operations that takes longer time. The long-running operation was performed in a new thread, which then reconnected or the result was be output to the main thread. In Android application, the concept there can only be one main thread and more than one worker threads. For the Services features to be used in application, a Service class has to be extended like this.

```
public class AlarmReceiver  extends  BroadcastReceiver {
     public void onReceive(context,intent){
     }
 }
```

Content Providers: Moving data around is essential when building Android application. This component takes care of supplying data from one application to another. Whenever a request is made to another application it could be SMS application, alarm application and email application. The Content providers handle the communication between the application and the calling application. GYMKit has made use of the gallery and the messaging applications in its implementation.

To use this property, the subclass must be extended.

```
public class MyContentProvider extends  ContentProvider {
      public void onCreate(){
      }
}
```

Broadcast Receiver: This component monitors the messages broadcasted from one application to another or from the system. This could be Alarm Broadcast, Battery Level or System Boot etc.
These messages were broadcasted and intercepted using the Intent Object and the BroadcastReceiver subclass like this.

```
public class MyReceiver  extends  BroadcastReceiver {
   public void onReceive(context,intent){
   }
}
```

5.3 Application Packages

The application packages consist of all fundamental features of the GYMKit application. The project was divided into three packages which was meant to make the project simple by separating concepts or components for easy interaction within similar components. All activity components are separated from the Bean or model components that serves as helper to the Activity implementation. The Adapter packages consist of all classes meant for custom screen adapter for all activity screen views. These are all needed in huge projects for better understanding of what is going on.

5.3.1 Activity Classes

The GYMKit project consist of 23 Activities. Some of the important ones are listed here and the implementation of the activities is listed in chapter 6. The following are few of the Activities;

DataLoaderSplash: This activity performs the function displaying a view on the screen that serves as a splash activity. The activity makes use of the loader_layout.xml for its content view layout. When this Activity is started, the Project name will be displayed for 3 seconds to the user and after the time out, the next Activity would be invoked.

The implementation has a package name activity. With this package name this activity is accessible throughout the whole project. This activity also extends the AppCompatActivity, which is the version 7 supports API for the basic Activity class.

Authentication1: This activity is tasked with the functionality of collecting the Email information of the user. This comes after the confirming that this user is a new user or has just installed the Application for the first time. This email is stored in the backend alongside with the user phone number for later Authentication.

Authentication2: This Activity is responsible for the second Authentication, which collects the user phone number by prompting user to enter the phone number in the space provided on the screen. The phone number is needed together with the user email. Both of them are used to create a profile for every user that install the GYMKit app. This Activity will also start the last authentication procedure of the whole project.

Authentication3: This is the final Activity that will authenticate the user if the user enters 5 digits which conforms with that contained in the backend. Every user granted access will have access to the Application main page which is named as HOMEPAGE.

Homepage Activity: The Homepage is responsible for displaying the following options:

1. Category
2. Settings
3. Favourite Workouts
4. Recent Workouts.

All the four options are displayed on the Navigation drawer. This Activity uses the Frame Layout to display each of the options displayed on the Navigation drawer.

Category: This extends the Fragment class. The Category Activity consists all the Categories of workouts available for Users. By default, the in-built or default Categories of workouts will be available for the user at the inception for the installation of the Application. After that, the user also possesses the ability to customize each of the categories by adding new workouts and performing every CRUD operation on them.

Settings: With this, the user could make some changes on the basic settings on the Application. The basics settings include changing user profile picture, profile name and Notifications.

Favourite Workouts: These are a selected list of workouts which are favoured by users.

Recent Workouts: This Activity list all recently performed workouts by user. As soon as the workout task is performed by the user, It will be tagged as being recent. This activity will only contains 10 workouts.

Description Activity: This is responsible for Displaying the image of the workout and contains the Description about the workout as well. The user can perform CRUD operation on this Activity.

Task: This Activity displays the fields of what is to be performed by every user of the App. The user can set default fields and Navigate to the description activity.

5.3.2 Model Classes

The Package consists of all classes which are needed by some of the classes of the Activity package. They function as the helping classes to the previously treated classes. In most cases, they help in the setting and getting of data within the package. The implementation of these model classes will be listed at the end of this section. The mainly used classes are mentioned below:

1. WorkoutBean.
2. DividerItemDecoration.
3. MyBinder
4. MyParcelable
5. NavigationDrawer
6. RoundImage

5.3.3 Adapter Classes

This package consists of classes which are adapter classes for most of the Activities used in the project. The implementation of the methods which are handled in the istener are made in the adapter classes. The two mostly used Adapter classes in the GYMKit project are WorkoutAdapter and the WorkoutCategoryAdapter. These two adapters contain most of the functionalities of both the workout and the category activity. The adapters also make use of the WorkoutBean in their operations. All the key features are made visible in their implementations.

5.4 Project Resources

GYMKit utilises various resources which are needed for the look and feel of the project. These resources contain folders consisting of files that are applied to all the project packages. The following are all the resources folders used in the GYMKit project; (Android Developer 2016)

1. Drawables: Bitmap files (.png, .9.png, .jpg, .gif).

2. Layout: XML files that define a user interface layout.

3. Menu: XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu.

4. Colors: XML files that define a state list of colours. They are saved in res/color/ and accessed from the **R. color** class.

5. Dimens: One of the XML files included in the values folder of the Android resource folder.

6. Strings: This resource provides text strings for an application with optional text styling and formatting.

7. Styles: This resource defines the format and look for a UI. A style can be applied to an individual view (from within a layout file) or to an entire activity or application (from within the manifest file).

6 GYMKit Application Testing

6.1 Testing Considerations

Testing a Mobile Application is a one of the Application procedures that encompasses the success of the application being developed before being deployed to the app store of Google play. Developers test for so many reasons such as battery drainage, application crash and poor performances. All the testing procedures are monitored for correction purposes. The performance of an application dictates what the application users do with apps. The security of the applications should also be scrutinized using sophisticated software.

The following are the basic things that were tested on the GYMKit application:

- Compatibility of the Scripts and the libraries.
- User Interface
- User experience tat includes navigation, help features, error messages and alert.
- Diverse layout UI views: Making an app more competitive in the market, developers should always ensure the development for different screen sizes of phones and tablets. Testers of the apps should majorly rely on emulators and stimulators that have certain limitations.
- User interaction: Mobile devices are becoming smarter, which give App designers a new testing procedure and components. Developers ensure app does not only work but also works well with the user interactions.
- Privacy and security: App developers should also test for the security of user data in the store.
- Mobile OS update: Mobile operating system mostly requires update. Developers should test for the efficiently in which the updates come in to play. Every update should be ensured for backward compatibility checking. Some apps depend on the old version of the API version and any updates must be ensured for compatibility or else the app would be rendered useless in the device

6.2 Application Testing Techniques

The mobile Application testing process includes the environment and the techniques that are used for the testing of the GYMKit app. The app is developed using the Android development environment, which is a tool containing the API level components. Specifically, the android version that was used for the development of the project is 5.1 and uses the API level 22.

The following are some of the testing procedures used for the testing of the GYMKit application:

- Using an automated testing with an emulator. The testing runs on an android device that is controlled by a computer using a USB cable with the presence of a drive in this case, a Google drive. The result can be captures using the performance tool.
- Database testing is also applied in the testing mechanism that is important for the confidentiality and integrity of the data. There is a need to check for the performance of the CRUD operation from the GYMKit app on the database, The Parse platform.
- Compatibility Testing is also made on the GYMKit app to ensure the application works as intended on the targeted devices. Targeted operating system, screen sizes and File system operations should work as planned.
- Functionality testing including testing for control widgets, storage media and Hardware performance. All these could be done manually. GYMKit app has its functionality testing done by running various operational methods several times.
- Power Consumption Testing was also made on the GYMKit app to check for any battery drainage operation. Wrong device settings could drain battery of a device and this is actually a concern for device app users. More and more testing approaches are carried out to stress test some methods with high power consumption.
- Usability Testing is one of the testing on the GYMKit app that ensures that the right user interface is used for its development This includes the Color and color contrast, UI widgets, widgets functionalities and navigation procedures.

7. Conclusion

The GYMKit project was developed for people who want to take full control of their body either through their workout instructors or through personal training plans. The project was successfully developed with categories of workouts and their respective descriptions. The users can perform routines contained in the GYMKit application and also have the opportunity of adding more training routines and workout procedures to the application.

The workout exercises have a strong and effective impact on the body muscles. It has been proving in so many ways to be efficient health-wise if well done. The efficient use of the application ensures that trainers will be able to monitor gradual stages in the development and changes of the body structures. Strict control is needed with the use of this application, which is the reason it includes a component where trainers and trainees have the options of determining whatever routine they would like to perform.

Since this is the first version of the project, more and more updates will be implemented later. I hope to add more features to the application in the future. Working hard to incorporate more components to the project and try to see other features needed by the application users. However, one component is the one where the app user would have the opportunity of sharing workout plans back and forth with their trainers in a certain format.

Lastly, it is encouraged that users of this application strictly utilize GYMKit with the guidelines of their workout trainers. This is very important for those who are under physio-therapist consultancy. Aside from this, I would strive to update the built-in workout routines and description and procedures on each workout based on regular practices. Later, the application will include a component that allows a trainer to add contact information on the application for those that need guidance on the workout procedures

References

Al Salool, Anas. (2012). *Android Architecture For System Application Software Stack.* http://android-app-tutorial.blogspot.fi/2012/08/architecture-system-application-stack.html (Accessed 8.22.2016).

Android Developer. (2016). https://developer.android.com/guide/topics/resources/available-resources.html (Accessed 5.28.2016).

Android. *SDK Platform Release Notes.* (2016). https://developer.android.com/studio/releases/platforms.html (Accessed 10.20.2016).

BodyBuilding. *Find A Plan.* (2016). http://www.bodybuilding.com/fun/find-a-plan.html (Accessed 7.12.2016).

Borodescu, Ciprian. (2013). *Web Sites vs. Web Apps: What the experts think.* https://www.visionmobile.com/blog/2013/07/web-sites-vs-web-apps-what-the-experts-think (Accessed 9.20.2016).

Continued Learning. (2015). “Native App Development and HTML5.” *Continued Learning.* http://blog.continued-learning.com/native-app-development-and-html5/ (Accessed 10.15.2016).

Dar, Hassan. (2015). *architecture of mobile software applications.* http://www.slideshare.net/hassandar18/architecture-of-mobile-software-applications (Accessed 10.12.2016).

Degges, Randall. (2015). *How to Manage API Authentication Lifecycle on Mobile Devices.* https://stormpath.com/blog/manage-authentication-lifecycle-mobile (Accessed 9.14.2016).

Dogtiev, Artyom. (2015). *App Usage Statistics: 2015 Roundup*. http://www.businessofapps.com/app-usage-statistics-2015/ (Accessed 10.8.2016).

Emis Group. (2016). *Physical Activity For Health*. http://patient.info/health/physical-activity-for-health (Accessed 10.1. 2016).

Idc. (2016). *Smartphone OS Market Share, 2016 Q2.* http://www.idc.com/prodserv/smartphone-os-market-share.jsp (Accessed 10.20.2016).

Kari, J.N. (2016). *Which Architecture Prevails for Mobile Application Development?* http://smartbridge.com/architecture-prevails-mobile-application-development/ (Accessed 10.5.2016).

Matt, K. (2011) *Native mobile app vs mobile web*. https://mlibraries.jiscinvolve.org/wp/2011/12/19/native-mobile-app-vs-mobile-web/. (Accessed 10.5.2016).

Meier, J.D. (2008) “Mobile Application Architecture Guide, Application Architecture Pocket Guide Series.” *Microsoft.* http://robtiffany.com/wp-content/uploads/2012/08/Mobile_Architecture_Guide_v1.1.pdf (Accessed 7 25, 2016).

Microsoft. (2016). *Chapter 3: Security Design Guidelines for Web Services.* https://msdn.microsoft.com/en-us/library/ff649737.aspx (Accessed 10.3. 2016).

Park, Alice. (2010). *Get Up! Sitting Less Can Add Years to Your Life.* http://healthland.time.com/2012/07/10/get-up-sitting-less-can-add-years-to-your-life/ (Accessed 10.22.2016).

Parse. *Android Guide.* (2016). http://parseplatform.github.io/docs/android/guide/ (Accessed 10.25.2016).

Rouse, Margart. (2016). *HCI (human-computer interaction)*. http://searchsoftwarequality.techtarget.com/definition/HCI-human-computer-interaction. (Accessed 8.20.2016).

Schweckel, Laura. (2014). *21 Low-Impact Workouts That Are More Effective Than You Think*. http://greatist.com/fitness/take-it-easy-21-unexpected-low-impact-workouts (Accessed 11.1.2016).

Scriptol. (2015). *Dalvik, the virtual machine of Android.* http://www.scriptol.com/programming/dalvik.php (Accessed 8.26.2016).

UK Department of Health. (2012). *Responsibility Deal network chair blogs about physical inactivity: the silent killer in our workplac.* https://www.gov.uk/government/news/responsibility-deal-network-chair-blogs-about-physical-inactivity-the-silent-killer-in-our-workplace (Accessed 10.20.2016).

Yarmosh, Ken. (2015). *How Much Does an App Cost: A Massive Review of Pricing and other Budget Considerations.* http://savvyapps.com/blog/how-much-does-app-cost-massive-review-pricing-budget-considerations (Accessed 10.15.2016).

Appendices

```
package activity;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;

import com.project.muftau.gymkit.R;

public class DataLoaderSplash extends AppCompatActivity {

    private static int LOAD_RATE = 3000;
    Context context= this ;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.loader_layout);

        new Handler().postDelayed(new Runnable() {

            @Override
            public void run() {
                SharedPreferences user =
context.getSharedPreferences("UserCheck", 0);
                String UserPhone =
user.getString("userPhone","");
                String UserDigit =
user.getString("userDigit","");
                if(UserPhone == ""){
                    Intent infoActivity = new Intent();

infoActivity.setClass(getApplicationContext(), Info.class);
                    startActivity(infoActivity);
                    finish();
                }
                else if(!UserPhone.isEmpty()){
                    Intent i = new
Intent(DataLoaderSplash.this,HomePage.class);
                    startActivity(i);
                    finish();
                }

            }
        }, LOAD_RATE);
    }
}
```

```java
public class Authentication1 extends AppCompatActivity implements AdapterView.OnItemSelectedListener {

    Button confirm;
    EditText EmailAddress;
    String contactLinkedAccount;
    String AuthDigit;
    boolean sent = false;
    private String userDigit = "";
    final Context context = this;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.authentication1);
        EmailAddress = (EditText) findViewById(R.id.authentication1_phoneNumber);
        confirm = (Button) findViewById(R.id.authentication1_continue);
        if(savedInstanceState !=null){
            EmailAddress.setText(savedInstanceState.getString("EmailAddress"));
        }


        confirm.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                //insert Email address into shared preference
                insertIntoSharedPreference();
            }
        });


    }
    public void insertIntoSharedPreference(){
        SharedPreferences sharedPrefs = getSharedPreferences("Existing_User", MODE_PRIVATE);
        SharedPreferences.Editor prefsEditor = sharedPrefs.edit();
        prefsEditor.putString("user_email", EmailAddress.getText().toString());
        prefsEditor.commit();
        Intent secondAuth = new Intent(Authentication1.this,Authentication2.class);
        startActivity(secondAuth);
        //finish();
    }
    @Override
    public void onSaveInstanceState(Bundle savedInstanceState) {
        super.onSaveInstanceState(savedInstanceState);
        // Save UI state changes to the savedInstanceState.
        // This bundle will be passed to onCreate if the process is
        // killed and restarted.
        savedInstanceState.putString("EmailAddress",EmailAddress.getText().toString());
    }

    public void onItemSelected(AdapterView<?> parent, View view,
                               int pos, long id) {
    }

    public void onNothingSelected(AdapterView<?> parent) {
        // Another interface callback
    }
```

```
package activity;

public class AddCategory extends AppCompatActivity {
    private static int RESULT_LOAD_IMAGE = 1;
    public boolean favourite = false;
    Bitmap SaveBitmap;
    Bitmap bitmapIcon;
    String picturePath;
    ParseFile file;
    Context context = this;
    ImageView categoryIcon;
    static final int[] dummyImages = new int[] {

R.drawable.gymkit,R.drawable.gymkit,R.drawable.gymkit,R.drawable.gymki
t };
    ImageView FavIcon,AddIcon;
    TextView FavText,AddText;
    TextView categoryName,targetMuscle,equipment;
    EditText categoryNameText,targetMuscleText,equipmentText;
    public RelativeLayout fav,add;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.add_category);
        Toolbar toolbar = (Toolbar)
findViewById(R.id.addCategoryToolbar);
        // Sets the Toolbar to act as the ActionBar for this Activity
window.
        // Make sure the toolbar exists in the activity and is not
null
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowTitleEnabled(false);

        FavIcon = (ImageView) findViewById(R.id.fav_cat_icon);
        FavText = (TextView) findViewById(R.id.fav_cat_text);

        fav = (RelativeLayout) findViewById(R.id.fav);
        add = (RelativeLayout) findViewById(R.id.add);

        AddIcon = (ImageView) findViewById(R.id.add_cat_icon);
        AddText = (TextView) findViewById(R.id.add_cat_text);

        // Textview widgets of the View
        categoryName = (TextView) findViewById(R.id.categoryName);
        targetMuscle = (TextView) findViewById(R.id.targetMuscle);
        equipment = (TextView) findViewById(R.id.equipment);

        // EditText Widgets of the View
        categoryNameText = (EditText)
findViewById(R.id.categoryName_editText);
       // categoryNameText.requestFocus();
        targetMuscleText = (EditText)
findViewById(R.id.targetMuscle_editText);
        equipmentText = (EditText)
findViewById(R.id.equipment_editText);

        targetMuscleText.setText(null);
        categoryNameText.setText(null);

        FavIcon.setImageResource(R.drawable.addtofav);
```

```
package activity;

public class AddWorkout extends AppCompatActivity {
    private static int RESULT_LOAD_IMAGE = 1;
    public boolean favourite = false,alarmState;
    Bitmap SaveBitmap;
    Bitmap bitmapIcon;
    String picturePath,WorkoutName;
    ParseFile file;
    Context context = this;
    ImageView categoryIcon;
    static final int[] dummyImages = new int[] {

R.drawable.gymkit,R.drawable.gymkit,R.drawable.gymkit,R.drawable.gymki
t };
    ImageView FavIcon,AddIcon;
    TextView FavText,AddText;
    TextView
Title,set,equipment,round,toolBarTitle,alarm,equip,description,muscle,
trainingText;
    EditText
Title_edit,set_editText,round_editText,equip_editText,description_edit
Text,muscle_editText;
    Switch myswitch;
    RadioGroup radioGroup;
    RadioButton radioButton;
    Button confirm,training;
    public String[] keys,values;
    public HashMap<String, String> trainingDays,dummyTrainingDays;
    public RelativeLayout fav,add;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.add_workout);
        trainingDays = new HashMap<String, String>();
        dummyTrainingDays = new HashMap<String, String>();
        Toolbar toolbar = (Toolbar)
findViewById(R.id.addCategoryToolbar);
        // Sets the Toolbar to act as the ActionBar for this Activity
window.
        // Make sure the toolbar exists in the activity and is not
null
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowTitleEnabled(false);
        toolBarTitle = (TextView) findViewById(R.id.toolbar_title);

        FavIcon = (ImageView) findViewById(R.id.fav_cat_icon);
        FavText = (TextView) findViewById(R.id.fav_cat_text);

        fav = (RelativeLayout) findViewById(R.id.fav);
        add = (RelativeLayout) findViewById(R.id.add);

        AddIcon = (ImageView) findViewById(R.id.add_cat_icon);
        AddText = (TextView) findViewById(R.id.add_cat_text);

        // Textview widgets of the View
        Title = (TextView) findViewById(R.id.Title);
        set = (TextView) findViewById(R.id.set);
        round = (TextView) findViewById(R.id.round);
```

```
package activity;

public class Authentication2 extends AppCompatActivity {
    EditText phoneNumberEditText;
    Button continueButton,cancelButton;
    String user_email,user_phoneNumber,subStr;
    private static int DELAY = 5000;
    private Handler handler;
    private ProgressDialog progress;
    private Context context = this;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.authentication2);

        phoneNumberEditText=
(EditText)findViewById(R.id.authentication2_digit);
        continueButton=
(Button)findViewById(R.id.authentication2_continue);
        cancelButton=
(Button)findViewById(R.id.authentication2_cancel);
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.SEND_SMS}, 1);
        continueButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                insertIntoSharedPreference();
                progress = new ProgressDialog(context);
                progress.setTitle("Please Wait!!");
                progress.setMessage("Wait!!");
                progress.setCancelable(false);

progress.setProgressStyle(ProgressDialog.STYLE_SPINNER);

                handler = new Handler() {

                    @Override
                    public void handleMessage(Message msg) {
                        progress.dismiss();
                        getDigit();

                        Intent Authentication3 = new
Intent(Authentication2.this, Authentication3.class);
                        startActivity(Authentication3);
                        super.handleMessage(msg);
                    }

                };
                progress.show();
                new Thread() {
                    public void run() {
                        handler.sendEmptyMessage(DELAY);
                    }

                }.start();

            }
        });
```

```
package activity;

public class Authentication3 extends AppCompatActivity {

    Button confirm;
    Button cancel;
    EditText AuthKey;
    private Handler handler;
    private Context context;
    private ArrayList<WorkoutBean> CategoryList = new ArrayList<>();
    private ProgressDialog progress;
    String UserObjectId;

    ArrayList<MyParcelable> myList = new ArrayList<MyParcelable>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.authentication3);
        context = this;
        AuthKey = (EditText)
findViewById(R.id.authentication3_authkey);
        confirm = (Button)
findViewById(R.id.authentication3_continue);
        cancel = (Button) findViewById(R.id.authentication3_cancel);

        confirm.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                //insert Email address into shared preference
                checkActualUserData();
            }
        });
        cancel.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                finish();
            }
        });




    }
    public void checkActualUserData(){
        SharedPreferences getsharedPrefs =
getSharedPreferences("Existing_User", MODE_PRIVATE);
        String prefAuthKey=
getsharedPrefs.getString("AuthKey","authentication key");
       // if(AuthKey.getText().toString().equals(prefAuthKey)){
            final ProgressDialog ringProgressDialog = new
ProgressDialog(Authentication3.this,R.style.TestTheme);
            ringProgressDialog.setCancelable(true);
            ringProgressDialog.setTitle("Please wait ...");
            ringProgressDialog.setMessage("Fetching all your workouts
...");
            ringProgressDialog.show();
```

```
package activity;

public class Category extends Fragment{
    private ArrayList<WorkoutBean> CategoryList = new ArrayList<>();
    private ArrayList<WorkoutBean> CategoryList1;
    private RecyclerView recyclerView;
    private WorkoutCategoryAdapter mAdapter;
    WorkoutBean category;
    String UserObjectId;
    private static int DELAY = 5000;
    private Handler handler;
    ParseFile file;
    //private Context context=;
    private ProgressDialog progress;
    float dummyfloat,thisdY;
    View itemView;
    Canvas dummyCanvas;
    int thisactionState;
    boolean thisisCurrentlyActive;
    int dummySwipePosition = 0,swipedPosition = -1;


    @Override
    public void onCreate(Bundle savedInstanceState) {
        mAdapter = new
WorkoutCategoryAdapter(CategoryList,getActivity());
        super.onCreate(savedInstanceState);
    }
    @Override
    public void onResume() {
        setUpItemTouchHelper();
        queryParseCategory();
        mAdapter.notifyDataSetChanged();
        super.onResume();
    }

    public void queryParseCategory(){
        //parseImageLoader();
        CategoryList.clear();
        SharedPreferences getsharedPrefs =
getActivity().getSharedPreferences("Existing_User",
Context.MODE_PRIVATE);
        String User_email = getsharedPrefs.getString("user_email",
"some email");

        ParseQuery<ParseObject> User = ParseQuery.getQuery("AppUser");
        User.whereEqualTo("email","tunde4mons@yahoo.com");
        User.getFirstInBackground(new GetCallback<ParseObject>() {
            @Override
            public void done(ParseObject object, ParseException e) {
                if (object != null) {
                    UserObjectId = object.getObjectId();
                    Log.i("myUser",UserObjectId);
                } else {

                }
            }
        });
        //ParseObject obj = ParseObject.createWithoutData("AppUser",
UserObjectId);
```

```
package adapter;

public class WorkoutCategoryAdapter extends
RecyclerView.Adapter<WorkoutCategoryAdapter.MyViewHolder>
        implements ItemTouchHelperAdapter {

    public ArrayList<WorkoutBean> CategoryList,itemsPendingRemoval;
    public ArrayList<WorkoutBean> dummy;
    Context context;
    Bitmap bitmapIcon;
    RoundImage roundImage;
    private static ItemTouchHelperAdapter HelperAdapter;
    boolean undoOn;
    View itemView;
    Dialog dialogButton;
    Dialog dialogHandler;

    public class MyViewHolder extends RecyclerView.ViewHolder
implements View.OnClickListener {
        public TextView categoryName, equipment, targetMuscle;
        public ImageView icon;
        //public ImageButton view_more;
        public Button undoButton;
        public ImageButton delete_button,edit_buton;
        public RelativeLayout item_row,undo_button_handler;


        public MyViewHolder(View view) {
            super(view);
            item_row = (RelativeLayout)
view.findViewById(R.id.item_row);
            undo_button_handler = (RelativeLayout)
view.findViewById(R.id.undo_button_handler);
            icon = (ImageView) view.findViewById(R.id.icon);
            categoryName = (TextView)
view.findViewById(R.id.categoryName);
           /* equipment = (TextView)
view.findViewById(R.id.equipment);*/
            targetMuscle = (TextView)
view.findViewById(R.id.targetMuscle);
            //view_more = (ImageButton)
view.findViewById(R.id.view_more);
            undoButton = (Button) view.findViewById(R.id.undo_button);
            delete_button = (ImageButton)
view.findViewById(R.id.delete_button);
            edit_buton = (ImageButton)
view.findViewById(R.id.edit_button);
            //view_more.setOnClickListener(this);
        }
        @Override
        public void onClick(View v) {

        }
    }


    public WorkoutCategoryAdapter(ArrayList<WorkoutBean> CategoryList,
Context context) {
        itemsPendingRemoval = new ArrayList<WorkoutBean>();
        this.CategoryList = CategoryList;
        this.context = context;
```