A Thesis/Project/Dissertation Report

on

ANDROID DEVLOPMENT OF MULTIFUNCTIONAL ANDROID APPLICATION

Submitted in partial fulfilment of the requirement for the award of the degree of

B.Tech CSE

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Name of Supervisor: Dr. Kavita
Designation : Associate Professor

Submitted By

UMAIR ALI
18SCSE1010483

NIKHIL YADAV
18SCSE1010287

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING /
DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
December, 2021

Table of Contents

List of Figures

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"ANDROID DEVLOPMENT OF MULTIFUNCTIONAL ANDROID APPLICATION"** in partial fulfilment of the requirements for the award of the "**Bachelors of technology in computer science and engineering**" submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of **August 2021** to **December 2021**, under the supervision of Dr. Kavita, Associate Professor, Department of Computer Science and Engineering of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project dissertation has not been submitted by us for the award of any other degree of this or any other places.

UMAIR ALI, 18SCSE1010483
NIKHIL YADAV, 18SCSE1010287

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Kavita
Associate Professor,
School of Computing Science and Engineering
Galgotias University, Greater Noida

CERTIFICATE

The Final thesis/Project/Dissertation Viva-Voce examination of UMAIR ALI, 18SCSE1010483 and NIKHIL YADAV, 18SCSE1010287 has been held on _____ and their work is recommended for the award of Bachelors of technology.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021

Place: GALGOTIAS UNIVERSITY, Greater Noida

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to several individuals and organizations for supporting me throughout our MAJOR PROJECT STUDY. First, we wish to express my sincere gratitude to our supervisor, Dr. Kavita, for her enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped us tremendously at all times in our research and writing of this REPORT. Her immense knowledge, profound experience and professional expertise in the field of computer science has enabled us to complete this research successfully. Without her support and guidance, this project would not have been possible. we could not have imagined having a better supervisor in our study.

We also wish to express our sincere thanks to “THE GALGOTIAS UNIVERSITY” for accepting us into their graduate program and giving us this golden opportunity, we are also grateful to all the university staff for their consistent support and assistance.

We would also like to show our deep gratitude to everyone in the SCHOOL OF COMPUTER SCIENCE AND ENGINEERING it was great sharing premises with all of you.

Thanks for all your encouragement!

ABSTRACT

In the last decades we have witnessed an enormous increase in the end user acceptance of mobile communications. The appearance of mobile platforms based on the open source software has rapidly increased the interest into mobile applications development. In this paper, we present an approach to the Android mobile phone application development that is based on an open source software and open source development environment. The work presented here is the outcome of a student project. We outline the experiences and lessons learned.

The mobile application development is an excellent choice for a beginner software engineering project that aims to introduce students to elementary development process activities such as design, implementation and testing. The reason is its low complexity, great opportunity for innovation, and huge market interest into new and innovative mobile applications. Due to its simplicity, the students can easily, and in relatively short period, get acquainted with elementary development concepts, techniques and resources that can be combined to produce mobile applications

Motivated by all these reasons, we have chosen to study the Android mobile phone application development and API Integrated Development Environment

Android OS is built upon Linux kernel, which allows Android to be ported to a wide variety of platforms by providing hardware abstraction layer for Android. It is used for memory management, process management, networking and other operating system services

The Real-time self-help, fitness and daily reminder application is a mobile application that acts as a central database and reminder application containing various attributes in it along with this it also provides and acts as a self-help and fitness trainer application. This android project is developed using android studio and Adobe XD as developing atmosphere along with use of languages java & XMI. it implements two major components of android application such as API and online database. A user using the application can see a wide range of features in respective categories. The user will also have to go through the login page in order to provide security this will be a very user friendly and user-oriented application

INTRODUCTION

The Android operating system is the largest installed base among various mobile platforms across the globe. Hundreds of millions of mobile devices are powered by Android in more than 190 countries of the world. It conquered around 75% of the global market share by the end of 2020, and this trend is growing bigger every other day. The company named Open Handset Alliance developed Android for the first time that is based on the modified version of the Linux kernel and other open-source software. Google sponsored the project at initial stages and in the year 2005,

it acquired the whole company. In September 2008, the first Android-powered device launched in the market. Android dominates the mobile OS industry because of the long list of features it provides. It's user-friendly, has huge community support, provides a greater extent of customization, and a large number of companies build Android-compatible smartphones.
As a result, the market observes a sharp increase in the demand for developing Android mobile applications, and with that companies need smart developers with the right skill set. At first, the purpose of Android was thought of as a mobile operating system. However, with the advancement of code libraries and its popularity among developers of the divergent domain, Android becomes an absolute set of software for all devices like tablets, wearables, set-top boxes, smart TVs, notebooks, etc.Google launched the first version of the Android platform on Nov 5, 2007. Since then, Google released a lot of android versions such as Apple Pie, Banana Bread, Cupcake, Donut, Éclair, Froyo, Gingerbread, Jellybeans, Kitkat, Lollipop, marshmallow, Nougat, Oreo, etc. with extra functionalities and new features
Programming Languages used in Developing Android Applications Java Kotlin Developing the Android Application using Kotlin is preferred by Google, as Kotlin is made an official language for Android Development, which is developed and maintained by JetBrains.

Previously before the Java is considered the official language for Android Development. Kotlin is made official for Android Development in Google I/O 2017.
Advantages of Android Development The Android is an open-source Operating system and hence possesses a vast community for support. The design of the Android Application has guidelines from Google, which becomes easier for developers to produce more intuitive user applications. Fragmentation gives more power to Android Applications. This means the application can run two activities on a single screen. Releasing the Android application in the Google play store is easier when it is compared to other platforms.
Disadvantages of Android Development Fragmentation provides a very intuitive approach for user experience but it has some drawbacks, where the development team needs time to adjust with the various screen sizes of mobile smartphones that are now available in the market and invoke the particular features in the application. The Android devices might vary broadly. So the testing of the application becomes more difficult. As the development and testing consume more time, the cost of the application may increase, depending on the application's complexity and features.

Android is a popular computing platform based on the Linux® operating system. The initial commercial version of Android hit the market in 2008 in the form of a mobile phone platform, back when the most popular cell phone for a business user was the BlackBerry, when the iPhone was beginning to make meaningful waves across all sectors, and when the majority of phone users were still tapping out texts from a flip phone.

Android has “paid its dues,” so to speak, in the smartphone market for the past decade. The success of Android and iPhone devices has rendered the one-time business mobile device market leader BlackBerry to be the subject of a Bruce Springsteen song: Glory Days. Interestingly, Android’s unprecedented success has helped push BlackBerry into a diverse set of offerings, including shipping devices running the Android platform. (Kudos to the BlackBerry team for pivoting and adding value to their shareholders and the broader market despite experiencing the retreat of their earlier dominance.)

In 10 years’ time, Android has effectively become the world’s most popular operating system by a number of measures. Despite the robust popularity of the flashy and capable Apple iPhone platform, Android shipments worldwide meaningfully outpace Apple’s offerings. While Apple’s devices continue to demand an ever-increasing price point, Android devices scale the global marketplace. Yes, there are super-pricy Android models sitting next to the latest iPhone, but there are also relatively low-cost Android phones and tablets available for sale at Walmart and on Amazon.

As Android has matured, it is finding its way into a variety of devices, including televisions, projectors, automobiles, and even recreational vehicles. Want to dim the lights in your camper or activate the awning? You can use the Android-based touchscreen interface to manipulate the controls. Or, use your smartphone equipped with Bluetooth to communicate with the RV’s Android-based control system. There are many of these types of interfaces finding their way to the market. Some user experiences are simply fantastic (like drone controllers), and some are less than fantastic, like the controls in my uncle’s RV. This article introduces the Android platform and discusses how you can use it for both mobile and non-mobile applications. The ambition is to get you on a path to making awesome apps for whatever platform arena you feel called to make your contribution.

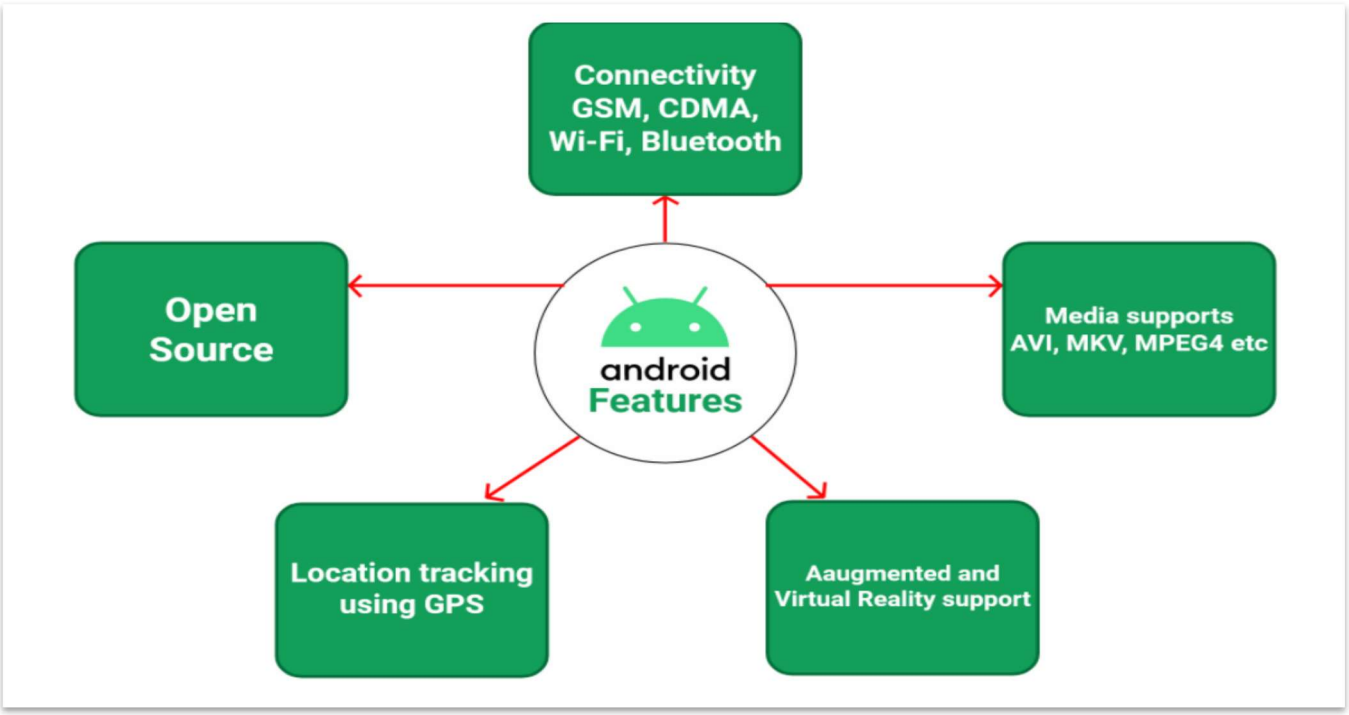

FIG. 1(ANDROID FEATURES)

The main objective of the project is to create an “REAL TIME SELF HELP AND DAILY UPDATE APPLICATION” that allows users with the functionality many applications inside one applications and

help user to keep in check of his daily activities while keeping in mind the fitness and health needs of an individual this application and so that the user can plan his day easily this application also have the perk of weather report to provide all this functionality this application implement two main features of android development that is API and online database this application will developed use android studio and adobe xd as development platform this app will have a very user centric and user friendly design

PROBLEM FORMULATION

Formulation of a research problem means to state the problem in a way that is researchable. It means to shape the research topic in a manner that it becomes ready for scientific investigation. A research problem is simply the research topic. A researcher needs to refine the topic and clearly state what is intended to be explored about the topic. This is called formulation of the research problem which involves narrowing down a broader research area into a specific research topic and devising the objectives. Once the research problem is formulated, the topic becomes ready for undergoing a scientific inquiry – *the research.*

The formulation of a research problem consists of the following steps:

1. Identify a broad research area of your interests:
2. Dissect the broad area into sub-areas:
3. Select one of the sub-areas
4. Raise research questions
5. Formulate the objectives

- **Identify a broad research area of your interest**

A researcher always starts with identifying a broad research area depending on his interest, knowledge, and expertise. It is generally a large area that a researcher wants to investigate. For instance, a researcher in social science may select research areas such as poverty reduction, overpopulation, conflict resolution, crime control, unemployment, political instability, economic fluctuations, human rights, the justice system, familial issues, cultural practices, social and religious fragmentations, domestic violence

- **Dissect the broad area into sub-areas**

Once a broad research area is selected, it is narrowed down into a specific topic that is researchable and manageable for the researcher. This involves dissecting the broad area into sub-areas and finding one suitable sub-area for the research.

- **Select one of the sub-areas**

As noted earlier, it is not feasible for a researcher to study all the sub-areas due to limited resources such as *time and monetary resources.* Similarly, one research study should specifically address one particular area so that it can be studied in its entirety. If a very broad area is selected, one research study may not do the justice in exploring its every aspect. This is because the boundaries of a large area are never easy to be identified in its entirety, and the researcher may unnecessarily pay more attention to some aspects while leaving other aspects of the research area.

- **Raise research questions**

After selecting a specific sub-area, the researcher has to think about what needs to be explored about this sub-area. In other words, the researcher has to raise questions related to the chosen sub-area which need to be answered through the research. Many research questions can be raised by the researcher, however, only the most important and relevant questions should be selected. The total number of questions (to be selected) depends on the nature of the topic which would also ultimately determine the overall length (or size) of the research thesis

- **Formulate the objectives**

Then, the researcher formulates the objectives of the research which are intended to be explored. These objectives basically stem from the research questions. The difference between research questions and objectives is the way they are written.

- **Common Challenges Android App Developers Face**

1. Hardware Features

The Android OS is unlike any other mobile operating system. For one thing, it is an open source system. Alphabet gives manufacturers the leeway to customize the operating system to their specific needs. Also, there are no regulations on the devices being released by the different manufacturers. As a result, you can find various Android devices with different hardware features running on the same Android version. Two smartphones running on Android latest ver, for example, may have different screen resolutions, camera, screen size, and other hardware structures. During android app development, developers need to account for all of this to ensure the application delivers a personalized experience to each user.

2. Lack of Uniform User Interface Design Rules

Since Google is yet to release any standard UI (user interface) design rules or process for mobile app developers, most developers don't follow any standard UI development rules or procedure. Because developers are creating custom UI interfaces in their preferred way, a lot of apps tend to function or look different across different devices. This diversity and incompatibility of the UI usually affects the user experience that the Android app directly delivers. Smart developers prefer to go for a responsive layout that'll keep the UI consistent across different devices. Moreover, developers need to test the UI of the app extensively by combining emulators and real mobile devices. Designing a UI that makes the app deliver the same user experience across varying Android devices is one of the more daunting challenges developers face.

3. API Incompatibility

A lot of developers make use of third-party APIs to enhance the functionality and interoperability of a mobile device. Unfortunately, not all third-party APIs available for Android app development are of high quality.. Some APIs were created for a particular Android version and will not work on devices running on a different version of the operating system. Developers usually have to come up with ways to make a single API work on all Android versions, a task they often find to be very challenging.

4. Security Flaws

As previously mentioned, Android is an open source software, and because of that, manufacturers find it easy to customize Android to their desired specifications. However, this openness and the massive market size makes Android a frequent target for security attacks. There have been several instances where the security of millions of Android mobile devices have been affected by security flaws and bugs like mRST, Stagefright, FakeID, 'Certifi-gate,' TowelRoot and Installer Hijacking. Developers need to include robust security features in their applications and utilize the latest encryption mechanisms to keep user information secure and out of the hands of hackers.

5. Search Engine Visibility

The latest data from Statista shows that Google Play Store contains a higher number of mobile apps. Additionally, a large number of Android users prefer free apps than paid apps which is why developers need to promote their mobile applications to increase their download numbers and employ application monetization options. The best way to promote the app to reach their target audience is to use comprehensive digital marketing strategies. Most developers make use of digital marketing professionals to promote their apps aggressively.

6. Patent Issues

Google doesn't implement any guidelines for the evaluation of the quality of new apps that are getting submitted to the Play Store. This lack of a quality assessment guideline causes a lot of patent-related issues for developers. Some developers, to avoid patent issues, have to modify and redesign their apps in the future.As per my personal experience, I have tried to cover general challenges faced by Android app developers. I'm sure keeping wary of these challenges would help developers to build successful apps in the most hassle free way.

TOOLS & TECHNOLOGY

- **SOFTWARE USED**

For the development of the project

- ANDROID STUDIO

Android Studio is Android's official IDE. It is purpose-built for Android to accelerate your development and help you build the highest-quality apps for every Android device. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug. Looking at Android's breadth of capabilities, it would be easy to confuse it with a desktop operating system. Android is a layered environment, one that is built upon a foundation of the Linux kernel and includes rich functionality. The user interface subsystem includes everything you would expect from a mature operating system environment including windows, views, and widgets for displaying common elements like edit boxes, lists, or drop-down lists. The browser is both capable for general web browsing and available for embedding directly into your own application.In the past decade, the mobile web has been transformed by the adoption of smartphones across consumer and business applications, including Android. "Responsive" web technologies have made the utility of mobile devices greatly enhanced

- ADOBE XD

Adobe is an American software company.Officially known as Adobe Systems, the company is known for its multimedia and creativity software products. Popular products include Photoshop, Acrobat Reader, and Adobe Creative Cloud.Headquartered in San Jose, California, the company was founded in 1982 by John Warnock and Charles Geschke. The name Adobe comes from Adobe Creek in California, which ran behind the houses of the company's founders.One of Adobe's first products was digital fonts, with the company entering the consumer software market in the 1980s. Adobe Illustrator was the company's first consumer product, which was a vector-based drawing program for Mac.Adobe XD is a powerful and easy-to-use vector-based experience design platform that gives teams the tools they need to craft the world's best experiences collaboratively. Available on Mac and Windows systems, XD meets teams where they're working with cross-platform compatibility.

Language tools used for development

- JAVA

Java is a programming language and computing platform first released by Sun Microsystems in 1995. It has evolved from humble beginnings to power a large share of today's digital world, by providing the reliable platform upon which many services and applications are built. There are many applications and even some websites that will not function unless you have Java installed. Java.com, this website, is intended for consumers who require Java for their desktop applications – specifically applications targeting Java 8. Developers as well as users that would like to learn Java programming should visit the dev.

- Xml

XML is a software- and hardware-independent tool for storing and transporting data. XML stands for extensible Mark-up Language it is a mark-up language much like HTML. it was designed to store and transport data, it was designed to be self-descriptive and it is a W3C Recommendation

- **HARDWARE USED**

Computer setup specification

- 8 gb RAM

At least 256 MB of RAM. The amount of RAM needed depends on the number of concurrent client connections, and whether the server and multiplexor are deployed on the same host and for suttel and staible development of project

- 1 tb SSD

Ssd is the recomended kind of drive for development because it isfast and helps in a seamless development of the project

- WINDOWS 11
- GTX 1050 Ti 4 GB
- INTEL i5 8 GEN

Technology used in development

- API

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.When you use an application on your mobile phone, the application connects to the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone. The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API.

To explain this better, let us take a familiar example.Imagine you're sitting at a table in a restaurant with a menu of choices to order from. The kitchen is the part of the "system" that will prepare your order. What is missing is the critical link to communicate your order to the kitchen and deliver your food back to your table. That's where the waiter or API comes in. The waiter is the messenger – or API – that takes your request or order and tells the kitchen – the system – what to do. Then the waiter delivers the response back to you; in this case, it is the food.

Here is a real-life API example. You may be familiar with the process of searching flights online. Just like the restaurant, you have a variety of options to choose from, including different cities, departure and return dates, and more. Let us imagine that you're booking you are flight on an airline website. You choose a departure city and date, a return city and date, cabin class, as well as other variables. In order to book your flight, you interact with the airline's website to access their database and see if any seats are available on those dates and what the costs might be.

The Modern API

Over the years, what an "API" is has often described any sort of generic connectivity interface to an application. More recently, however, the modern API has taken on some characteristics that make them extraordinarily valuable and useful:

- Modern APIs adhere to standards (typically HTTP and REST), that are developer-friendly, easily accessible and understood broadly
- They are treated more like products than code. They are designed for consumption for specific audiences (e.g., mobile developers), they are documented, and they are versioned in a way that users can have certain expectations of its maintenance and lifecycle.
- Because they are much more standardized, they have a much stronger discipline for security and governance, as well as monitored and managed for performance and scale
- As any other piece of productized software, the modern API has its own software development lifecycle (SDLC) of designing, testing, building, managing, and versioning. Also, modern APIs are well documented for consumption and versioning.

- **ONLINE DATABASE**

An online database is a database accessible from a local network or the Internet, as opposed to one that is stored locally on an individual computer or its attached storage (such as a CD). Online databases are hosted on websites, made available as software as a service product accessible via a web browser. They may be free or require payment, such as by a monthly subscription. Some have enhanced features such as collaborative editing and email notification.

Cloud database

A cloud database is a database that is run on and accessed via the Internet, rather than locally. So, rather than keep a customer information database at one location, a business may choose to have it hosted on the Internet so that all its departments or divisions can access and update it. Most database services offer web-based consoles, which the end user can use to provision and configure database instances.

LITERATURE SURVEY

In the recent years, the advances in mobile technology have brought an exorbitant change in daily lifestyle of individuals. Smartphones/mobile devices are rampant in all aspects of human life. This has led to an extreme demand for developing software that runs on mobile devices. The developers have to keep up with this high demand and deliver high-quality app on time and within budget. For this, estimation of development and testing of apps play a pivotal role. In this paper, a Systematic Literature Review (SLR) is conducted to highlight development and testing estimation process for software/application. The goal of the present literature survey is to identify and compare existing test estimation techniques for traditional software (desktop/laptop) and for mobile software/application. The characteristics that make mobile software/application different from traditional software are identified in this literature survey. Further, the trend for developing the software is towards agile, thus this study also presents and compares estimation techniques used in agile software development for mobile applications. The analysis of literature review suggests filling a research gap to present formal models for estimating mobile application considering specific characteristics of mobile software.

The mobile devices being utilitarian, user-friendly, accessible has made it the most popular and indispensable expedient for human essentials from the past few years (Malavolta et al., 2015). Mobile software developers' are driven to release software on time and within budget. Software estimation plays a pivotal role in providing the most accurate sizing figure for building confidence in developers and stakeholders relationship (Soares and Fagundes, 2017). Many approaches used for estimation of traditional software are adapted for mobile application development and testing (Wasserman, 2010).The testing phase of traditional software development proceeds through additional life cycle called Software Testing Life Cycle (STLC) (Katherine and Alagarsamy, 2012). According to Gao et al. (2014) mobile software testing are set of activities for mobile apps on mobile devices by exhausting definite software test techniques and tools in order to confirm quality in functionality, performance, and QoS, as well as features, like mobility, usability, interoperability, connectivity, security and privacy.

The main phases of the testing process include test planning, test designing, test execution and test analysis (Farooq et al., 2011, Amen et al., 2015).

The estimation of effort for software testing comprises an estimation of test size, effort (Person per Hour), cost and entire schedule by means of several methods, tools and techniques (Abhilasha and Sharma, 2013). If effort, time and cost required to test the software can be anticipated, the testing resources can be systematically planned within a set target date to ensure lucrative culmination of projects. According to Zhu et al. (2008b), for estimating the test effort the major consideration is given on test designing (creation of test cases) and test execution.

With the advent of Agile Software Development (ASD) (Usman et al., 2014) entire software development community has been driven by the adoption of agile methodology. The Agile approach to mobile application development states an iterative and incremental approach comprising self-organizing teams and cross-functioning teams working together to build the software (Kaur, 2016). The prominent existing agile mobile application development approaches are MOBILE-D, RaPiD7, Hybrid methodology, MASAM, Scrum with Lean Six Sigma (SLeSS) (Dewi and Nur Atiqah Sia, 2015). The Agile espousal to mobile application development is considered as a natural fit by many researchers (Cunha et al., 2011, Rahimian and Ramsin, 2008, Scharff and Verma, 2010). In an agile environment, development and testing are not considered separate phases as in traditional software development (Rahimian and Ramsin, 2008). The estimation of software in agile is prepared for both development and testing together. Estimation of effort in agile development is a new area of focus and very less work is reported literature (Aslam et al.,

2017).

Another major contribution is identifying the characteristics of mobile apps that make them distinct from traditional software.Subsequently, the paper is divided as follows: Section 2 presents the research method comprising three phases of Systematic Literature Review (SLR). First and second phase is devoted to forming Research Questions (RQ) and finding relevant literature for studies. The results of the review are analyzed in the third phase i.e. result reporting phase of SLR, answering each Research Question (RQs). In section 3, discussions, research gaps and future directions are presented. Some threats to the validity of SLR are discussed in section 4 followed by conclusions in Section 5.

This section outlines the related literature and findings by the researchers which form the desired background for this research. The guidelines provided by Kitchenham and Charters (2007) are followed by conducting Systematic Literature Review (SLR). SLR is a research manner for carrying out a literature review in an orderly way of charting definite phases. SLR method uses three phases for performing literature review including Planning and specifying research questions, conducting the review that comprises an identification of search string and data sources, selecting studies, quality assessment, and data extraction and finally reporting the review. The steps followed for systematic literature review are undertaken in the following sections of this paper. The overview of the systematic literature review is shown in Fig. 2.

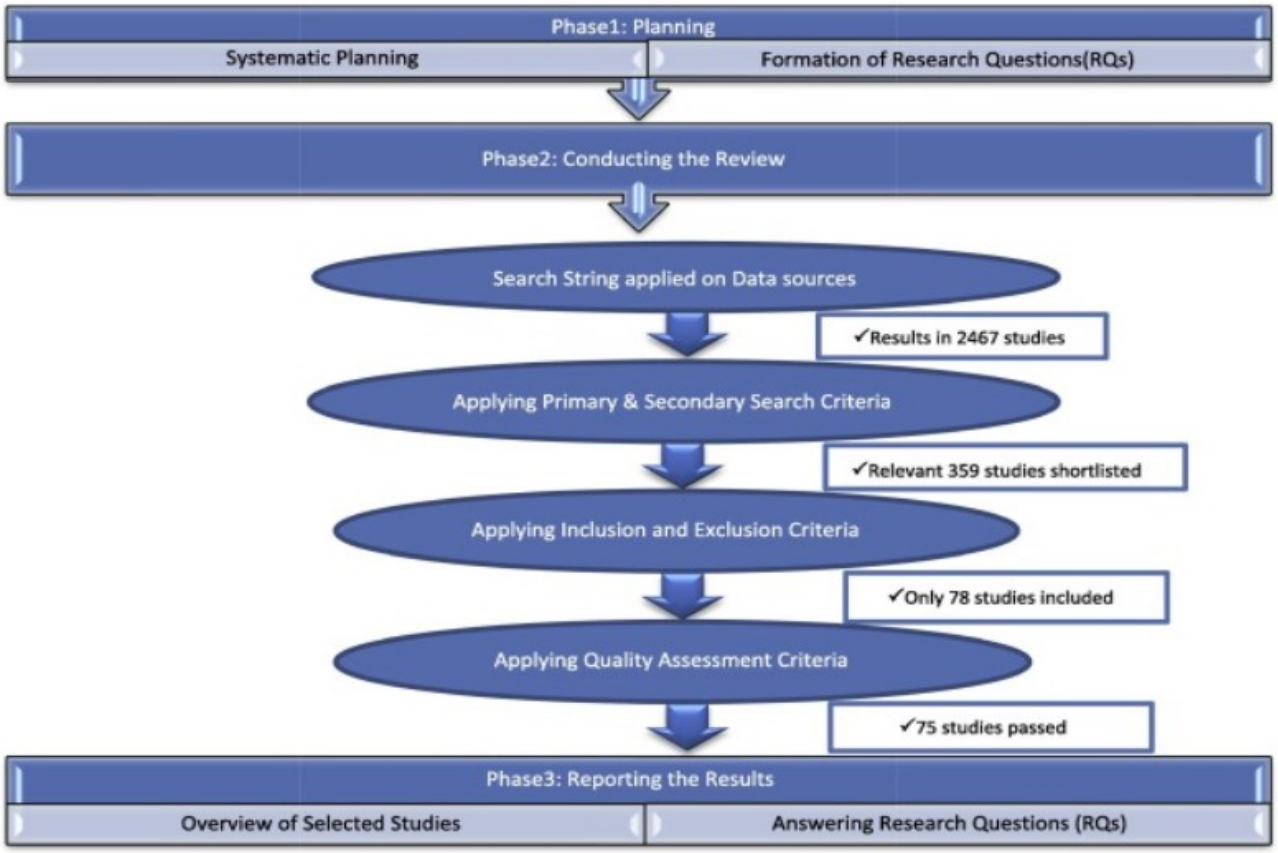

Fig. 2. (overview of the survey)

For the smooth conduct of systematic literature review, proper planning is fundamental for smooth execution of SLR. The research questions derive from the entire systematic literature review planning phase.

Affirming the research questions is the vital part of any systematic review. In accordance with guidelines

proposed by Petticrew and Roberts (2006) the criteria to frame research questions are based on PICOC (Population, Intervention, Comparison, Outcomes, and Context). If the research question is not outlined properly, the literature review may turn out off course. For this study, PICOC is defined as shown below

PICOC with description.

PICOC	Description
Population	Mobile Application projects.
Intervention	Test Effort estimation techniques/methods/process.
Comparison	Traditional software test effort estimation techniques with mobile apps testing estimation.
Outcomes	Mobile software test estimation techniques and characteristics of mobile apps that are considered important in development and testing estimation.
Context	Review the existing studies on test estimation of mobile Apps.

Research questions (RQs)

The review questions steer the entire systematic review methodology. The foremost aim of the review is to answer the following research question:-

RQ1. What are currently known software test estimation techniques for traditional applications?

RQ2. What are mobile development and testing estimation techniques?

This RQ can be subdivided into two sub-categories:-

RQ2.a. What are mobile development and testing estimation techniques in a traditional software development environment?

RQ2.b. What are mobile development and testing estimation Techniques in agile software development?

The data extraction phase elaborates the mining of data from the final selected studies that address the peculiarities of RQs. The data extraction for the finally chosen studies are done in an MS Excel sheet.

This section defines the results relayed to the systematic literature review questions. The results are presented in a tabular format for each study.

- **Selected studies overview**

The distribution of the chosen studies through the published sources. Out of the 75 studies, 24(32%) came from IEEExplore, 11 studies (14%) came from SpringerLink, ACM Digital Library 14(19%), 11(15%) from Research Gate, 3(4%) studies from CiteSeer, 2(3%) from Elsevier ScienceDirect, 3(4%) from InderScience and 7(9%) from others (ProQuest, GoogleScholar, TU/e Repository, scielo.org.co, SemanticScholar, IGI Global). The distribution of selected studies from different sources is shown in . Maximum papers are referred to the year 2014, 2015, 2016 and one each from 1999, 2001 and 2005. The distribution of selected studies according to the published year

- **Results reporting on RQ1**

To answer RQ1, out of seventy-five selected studies, twenty-six studies cover all the facets of RQ1.

Test Point Analysis (TPA) proposed by Veenendaal et al. (1999) is based on function point analysis used for estimating the functional size of software with additional attributes such as testing strategy and productivity.
Use Case Point Analysis (UCP) by Nageswaran (2001) examines testing characteristics and their complexity along with software development complexity.
Test Execution Points is a technique proposed by (E Aranha and Borba, 2007a, Aranha and Borba, 2007; Eduardo Aranha and Borba, 2007) based on test specification. The test cases are assigned execution points and then the effort is calculated
The model proposed by Abran et al. (2007) for estimating the test volume and effort, the functional requirements are taken as bases to form test estimation and then nonfunctional requirements are taken into consideration.
An approach by Kushwaha and Misra (2008) cognitive information complexity calculation and McCabe's Cyclomatic complexity measure is used to estimate the test execution effort.
Another approach by Zhu et al. (2008a) consist of three attributes for test effort estimation namely test case number, test execution complexity, and tester and then uses a historical database to assign effort.
Another approach by Zhu et al. (2008b) is an extension to existing UCP and considers test execution as a two-dimensional vector having testing experience and knowledge of the application.
The method suggested by Lazić and Mastorakis (2009) covers white box testing and test activities based on Software/System Test Point (STP) metric. The model is implemented on estimating object-oriented software projects by computing size and then applying three steps of COTECOMO model.
A model presented by Silva et al. (2009) is based on historical efficiency data of testers and functional test execution effort estimation and then the model accuracy is measured against different software as a case study.
Another model presented by Abhishek et al. (2010) studies the use of use case and neural network in Precoding phase and then in postcoding phase again neural network with variable, complexity and criticalness component as input is used to calculate test efforts.
In the model presented by Souza and Barbosa (2010) modified TPA is used by making it simpler and hence easy to use. In this model, there are two steps:-one followed by the test designer and second by the tester. Each of the steps has further sub-steps to follow to finally provide test effort estimation.
Aloka et al. (2011) presented an approach which is a combination of UCP, TPA, and particle swarm optimization (PSO) based approach to optimize the test effort estimation.
The approach proposed by Srivastava et al. (2012) is an adaptable model of UCP along with cuckoo search, a *meta*-heuristic approach, for test effort estimation.
A model is proposed by Sharma and Kushwaha (2013) based on SRS, then the complexity of requirements is computed. Requirement based test function points (RBTFP) is calculated based on the complexity of requirements and Technical and Complexity Factors (TEF).
Bhattacharya et al. (2012) proposed an approach which considers features of the software testing effort (STE) estimation by proposing a soft computing technique, Particle Swarm Optimization (PSO) along with COCOMO and test effort drivers into a single stage.
In the approach by Nguyen et al. (2013) the test case point is derived by measuring checkpoints, preconditions, test data and type of testing. The authors compared the proposed approach on two industrial case studies that used the experienced-based approach of testers.
Another heuristic approach by Srivastava et al. (2014) based on bat algorithm used along with existing test effort estimation techniques, UCP and TPA. Later the results obtained after applying bat algorithm are compared with those obtained from UCP and TPA to conclude that findings are improved and nearer to the actual effort using bat algorithm.
In the model proposed by Zapata-Jaramillo and Torres-Ricaurte (2014), the concept of a pre-conceptual schema is used which is a graphical technique to show domain knowledge in a natural language.
An approach by Hauptmann et al. (2014), the test suits are taken as an input and then a cost model is designed based on estimation done by an expert. In the cost model estimation is provided for test suite creation, test suite maintenance, and test execution.

The model presented by Srivastava (2015) uses fuzzy logic and fuzzy multiple linear regression techniques along with COCOMO-II to estimate software test effort. The problem with the model is usability while designing fuzzy rules. However, results produced using this model are better than existing methods.
The method proposed by Arumugam and Babu (2015) is based on the UCM (Use Case Model) and Function Point Analysis (FPA). The use case model is adapted to use case graph and later the edges acting as alternatives for required components are assigned weights. Then FPA is followed for assigning appropriate complexity weights to System Testing Technical Complexity Factors.
An approach by Badri et al. (2015) is used in a model that covers unit testing effort only and forms its bases on Quality Assurance Indicator (Qi) metric along testing effort comprised in scripting unit test cases.
An automatic tool, PredSym, is presented by Bhattacharyya and Malgazhdarov (2016) predicts the code coverage for testing by using a machine learning technique called symbolic execution tool i.e., KLEE.
Islam et al. (2016) demonstrated a web-based test effort estimation tool based on COCOMO-II and have successfully implemented it on 22 projects.
Jin and Jin (2016) proposed an optimization technique called quantum particle swarm optimization (QPSO) algorithm for optimizing parameters in test effort function (TEF) used in Software Reliability Growth Model.
Table 7 lists the summarized review in form of a matrix of all studies selected for studying test effort estimation techniques in traditional software. From the table 7, it can be analyzed that model-based approaches are prominently followed in most of the studies. The tool support for estimation is found hardly in 4 studies. Some studies have only proposed a model and have not validated it on industrial projects.

- **Results reporting on RQ2**

The focus of this research is on mobile applications rather than on traditional applications, RQ2 focuses on elaborating estimation of development and testing of mobile apps in traditional development process and Agile Development process. Out of seventy-five selected studies, twenty-two studies are ardent to answer RQ2.

- **raditional techniques for estimating mobile application development and testing**

Seventeen studies out of the selected twenty-two investigated the traditional estimation techniques for mobile applications. There are many development estimation techniques and many testing effort estimation techniques in literature for traditional software. But as the focus is on mobile applications, this study covers development effort and test effort for only mobile software. Table 10 lists the identified techniques where an agile methodology is not followed for the development of mobile apps. The techniques are broadly classified into three categories by Mendes (2007) i.e. Algorithmic-based models, Expert Judgment based models, and analogy based models. Some of the approaches consider estimation of development and testing of the mobile app as a single process and two studies have considered test estimation of mobile apps as a separate one. COSMIC Function Size Measurement (Abdullah et al., 2014, D'Avanzo et al., 2015, de Souza and Aquino, 2014, Ferrucci et al., 2015, Heeringen and Gorp, 2014, Nitze, 2013, Sellami et al., 2016, Vogelezang et al., 2016) is frequently used for estimation technique which is used to measure functional size of the mobile app. Other types of estimation techniques identified are Function Point Analysis (Preuss, 2013, Tunalı, 2014) and Use Case Point (Haoues et al., 2017) which are algorithmic-based models that measure functional, technical factors and environmental factors for estimation. Regression-Based technique (Shahwaiz et al., 2016) uses a parametric model based on effort predictors and data points collected through an online questionnaire which are further used in the regression model. Delphi method (Catolino et al., 2017) is based on experience to estimate the effort whereas Architecture Based estimation model (Wadhwani et al., 2008) for reliability and testing estimation of the mobile application is proposed and the case study was conducted in two companies.
Another algorithmic approach for estimating the cost of developing Android mobile apps are based on COCOMO –I and II model (Asghar et al., 2016). Analogy-based estimation plus functional size measurement (Nitze et al., 2014) approach is also proposed for mobile apps. One approach (E Aranha and

Borba, 2007b, Aranha and Borba, 2007) covers estimation of test execution effort taking a risk and test factors as a major contributing feature in estimation and taken mobile application as a case study.

- **Agile techniques for estimating mobile application development and testing**

The agile methodology aims at facilitating software development processes where changes are acceptable at any stage and provide a structure for highly collaborative software development. In such a dynamic environment, estimation is very challenging (Usman et al., 2014).
Agile approach to mobile application development estimation has very less number of studies. One of the reason could be the adoption of agile to mobile context is still in its evolving phase. The identified studies are listed in Table 11. It can be seen that only one study has proposed a technique for test effort estimation for the mobile app in an agile environment. Other studies consider development and testing estimation together as a single practice.

Traditional use case point method of estimation is extended by adding efficiency and risk factor of testers in the agile team (Parvez, 2013). Another technique (Francese et al., 2015) is based on a stepwise linear regression model which estimate the effort for Android apps from requirements specification including a number of use cases, actors, etc. User story point (Aslam et al., 2017) is refined by considering additional factors along with size and complexity. The quality factor, Novelty factor and Type factor of User Story are added to deliver the best estimation of mobile application development. Additional approach (Qi and Boehm, 2017) uses Early Use Case Point (EUCP) and Extended Use Case Point (EXUCP) along with COCOMO drivers at different iteration levels in agile mobile app development. An experience-driven approach using Delphi technique (Lusky et al., 2018) is used for effort estimation in which mobile app is taken as case studies.

- **Results reporting on RQ3**

The results from Systematic Literature Review (SLR) recognized 15 characteristics in the majority of the chosen studies after passing all the selection criteria from primary studies. Twenty-seven studies are dedicated to answering RQ3 out of the total seventy-five selected studies. Some of the characteristics are deliberated as characteristics of the mobile device (Limited RAM, Battery, Memory, and Screen size) however many studies emphasize that they need to be considered as these limitations are directed linked to mobile apps development and testing. Fig. 5 shows the type of mobile app characteristics mentioned in selected studies. Table 14 lists the studies in which each characteristic is discussed. The findings of this SLR for RQ3 clearly state that how these mobile app characteristics are different from traditional software. So the development and testing estimation techniques reported in RQ1 and RQ2 does not consider these important characteristics undertaking the estimation process.

- **Description of mobile app characteristics**

- Limited Memory: - The internal memory of the mobile device is limited. The mobile app consumes a memory space when it is installed on the device. The developers should use such programming practices that allow development of small size apps. The testers should check how the app performs when the memory of the device reaches maximum memory limit (Amalfitano et al., 2011, Cao et al., 2012, Charland and Leroux, 2011, Dantas et al., 2009, Kim et al., 2009, Kim, 2012, Liu et al., 2014, Lu et al., 2012, Muccini et al., 2012, Vilkomir and Amstutz, 2014, Zein et al., 2016, Zein et al., 2015).

- Limited CPU or Small Processing capacity: - As the mobile devices have small processors, the mobile apps should be developed and tested in a way so as to decipher the consumption of the processor while it runs on the mobile device (Amalfitano et al., 2011, Cao et al., 2012, Charland and Leroux, 2011, Ciman and Gaggi, 2017, Dantas et al., 2009, Kim, 2012, Liu et al.,

2014, Muccini et al., 2012, Nidagundi and Novickis, 2017, Zein et al., 2016, Zhang and Adipat, 2005).

- Limited RAM: - Apps should be programmed and tested so that they exhaust less amount of memory when they run on the mobile device. Large size mobile apps tend to run slow and further influence user experience (Cao et al., 2012, Charland and Leroux, 2011, Ciman and Gaggi, 2017, Kim, 2012, Liu et al., 2014, Lu et al., 2012, Muccini et al., 2012, Nidagundi and Novickis, 2017, Zein et al., 2016).

- Limited screen size and Orientation: - Mobile devices have a small screen. Keeping the constraint in mind the app should be developed and tested well to check if it operates differently on varied screen size and orientation (Arzenšek and Heričko, 2014, Costa et al., 2014, Holl and Elberzhager, 2016; M. Amen et al., 2015, Nidagundi and Novickis, 2017, Vilkomir and Amstutz, 2014, Zhang and Adipat, 2005).

- Limited Battery: – Mobile devices have very limited battery life. The mobile apps should be developed in a way so they should consume less battery power. The app should be tested in a scenario when the battery is too low that how it behaves in this instance and should retain data integrity when the battery dies (Amalfitano et al., 2011, Cao et al., 2012, Dantas et al., 2009, Kim, 2012, Liu et al., 2014, Lu et al., 2012, Muccini et al., 2012, Nidagundi and Novickis, 2017, Zein et al., 2016, Zhang and Adipat, 2005).

- The diversity of User interfaces (touchscreen, keypad, and voice):- As input to a mobile device can be through voice, touch keypad, stylus, etc., the mobile app should be tested against all input interfaces (Arzenšek and Heričko, 2014, Charland and Leroux, 2011, Costa et al., 2014, de Cleva Farto and Endo, 2017, Kim, 2012, Kirubakaran and Karthikeyani, 2013, Liu et al., 2014, Muccini et al., 2012, Zein et al., 2016, Zein et al., 2015, Zhang and Adipat, 2005).

- Context awareness: - Mobile apps can react variedly based on their environment which means that the app should be tested to take into account all the input explicitly delivered by operators and likewise implicit input regarding physical and computational context of operators (Arzenšek and Heričko, 2014, Charland and Leroux, 2011, Ciman and Gaggi, 2017, Holl and Elberzhager, 2016, Kirubakaran and Karthikeyani, 2013, Muccini et al., 2012, Nidagundi and Novickis, 2017, Zein et al., 2016, Zhang et al., 2015).

- Diverse Mobile Connections (2G, 3G, 4G and various wireless networks), Mobile network operators and user's mobility: – Mobile app should be tested under all different connections such as Wireless networks, Bluetooth, 3G, 4G, NFC, etc., (Arzenšek and Heričko, 2014, Charland and Leroux, 2011, Dantas et al., 2009, Franke et al., 2012, Giessmann et al., 2012, Göth, 2015, Kim et al., 2009, Kirubakaran and Karthikeyani, 2013, Lu et al., 2012, Muccini et al., 2012, Nidagundi and Novickis, 2017, Zein et al., 2015, Zhang and Adipat, 2005).

- Different application types (Native, Hybrid, and Web):- The development and testing of native, web and hybrid mobile application is different. So each one should be tested thoroughly depending on the type of app (Cao et al., 2012, Charland and Leroux, 2011, Dantas et al., 2009, Giessmann et al., 2012, Kim et al., 2009, Liu et al., 2014, Lu et al., 2012, Muccini et al., 2012, Nidagundi and Novickis, 2017, Vilkomir and Amstutz, 2014, Zein et al., 2015).

- Diverse operating systems (software):- The mobile apps run on the particular operating system. There are various mobile OS such as iOS, Android, RIM, Windows, and Symbian etc. The app should be tested for the required platforms for proper compatibility (Ciman and Gaggi,

2017, Giessmann et al., 2012, Kim, 2012, Lu et al., 2012, Amen et al., 2015, Muccini et al., 2012, Nidagundi and Novickis, 2017, Umuhoza and Brambilla, 2016, Vilkomir and Amstutz, 2014, Zein et al., 2015, Zein et al., 2016, Zhang et al., 2015).

- Diverse devices (hardware):- Mobile devices get launched in the market every now and then with a change in technology. The mobile App should be tested for maximum no. of devices wherever possible (Ciman and Gaggi, 2017, Franke et al., 2012, Kim et al., 2009, Kim, 2012, Kirubakaran and Karthikeyani, 2013, Lu et al., 2012, Amen et al., 2015, Nidagundi and Novickis, 2017, Vilkomir and Amstutz, 2014, Zein et al., 2016, Zhang et al., 2015).

- Interrupt: -The mobile app should be tested for all kind of interruptions such as receiving a message, battery low, in between calls; while it is running on the mobile device (Charland and Leroux, 2011, Dalmasso et al., 2013, de Cleva Farto and Endo, 2017, Nidagundi and Novickis, 2017, Umuhoza and Brambilla, 2016).

- Integration with other Apps: - There are some apps that run in integration with other apps. Testing should be done to check if mobile app integrates well with other apps on the user's device or not (Charland and Leroux, 2011, Dantas et al., 2009, Giessmann et al., 2012, Muccini et al., 2012, Umuhoza and Brambilla, 2016).

- Network Availability: - Network availability varies, so apps should be developed and tested keeping this constraint in mind. It should be tested how it behaves when the user moves to the remote area when networks are not in range (Arzenšek and Heričko, 2014, Dantas et al., 2009, Franke et al., 2012, Göth, 2015, Holl and Elberzhager, 2016, Kim et al., 2009, Kirubakaran and Karthikeyani, 2013, Muccini et al., 2012, Nidagundi and Novickis, 2017, Zein et al., 2015, Zhang et al., 2015)

- Response Time: - The mobile app should be tested for its start time which should be immediate through any input interface means (Cao et al., 2012, Dantas et al., 2009, Kim et al., 2009, Liu et al., 2014, Nidagundi and Novickis, 2017, Vilkomir and Amstutz, 2014, Zein et al., 2015).

- **Discussion, research Gap, and future work**

The results from SLR for answering RQ1 indicate that the model-based approach is followed by many researchers. The base input to most of the traditional test estimation techniques is functional requirements that are then used to derive functional size. Use Case Point; its extension and optimizations are prominently exploited among all the identified techniques in traditional software test estimation. The tool support for test effort estimation is very limited. In order to measure the accuracy of the estimation techniques, the estimated effort is compared to the actual effort. Apart from this, the other statistical measure MRE and MMRE are also widely accepted.

Answers for RQ2 regarding estimation techniques for development and testing of mobile apps are identified both in traditional software development environment and agile software development. In mobile applications testing estimation algorithmic-based models are prevalent. COSMIC FSM techniques are preferred due to its designing in such a way that it could be applied in a very broad range of architectures, including mobile apps. COSMIC FSM provides the functional size of the software which is used to derive effort for estimation of mobile app development. Majority of reported studies are contributed towards estimation on the development and for testing effort estimation, only two studies are reported. According to Jayakumar and Abran, (2013), COSMIC size-based estimation models can be used to estimate efforts for the full life cycle of software development or any major life cycle activities such as

Testing. This can serve as a future direction when instigated in the mobile domain for proposing a standardized model and validate the estimation results of the model on mobile apps.
Adoption of agile to mobile context is still in its evolving phase. From Section 2.3.3.2 it can be perceived that very less number of studies is proposed in test effort estimation for the mobile app in an agile environment. Algorithmic-based models are reported mainly in three identified studies. The main attribute in the estimation of mobile app reported in maximum identified studies is size. The size is measured in terms of cosmic function point, Function point, use case point, and user story point. For measuring the accuracy of estimation models in mobile app domain, MMRE and Pred(x) are reported by most studies.
In answer to RQ3, fifteen characteristics are identified in section 2.3.4. Testing of the mobile app on "different mobile OS" is identified in maximum studies along with testing on "different mobile connections". "Limited memory" of mobile devices which is rather a mobile device constraint is also identified as mobile app characteristic in many studies as how much memory a mobile app consume while running on the device poses a testing constraint too. Other characteristics are discussed in Section 2.3.4.1. As for estimation of testing of the mobile app, these identified characteristics may or may not affect the test estimation process. The impact of each characteristic while performing test estimation can range from being negligible to highly significant. A survey on investigating the impact of mobile app characteristics, accumulated from mobile app developers and testers can be beneficial to accomplish this task. This identified research gap can be considered as probable research direction for future work.

- **Threats to validity**

The validity threats for SLR are discussed in this section. The *construct validity threat* in terms of SLR (Systematic Literature Review) is its failure to claim coverage of all relevant studies. By adopting a good search strategy and using all relevant search strings and their synonyms we tried to mitigate this threat. Also, only one study each from year 1999 and 2001 is selected as they represented major techniques for test effort estimation in traditional software which are further enhanced and modified by other authors. Rest of the selected studies for reporting test effort estimation in traditional software is based from year 2007 till 2016 i.e. last decade.
Internal Validity threat concerns with data extraction phase of SLR. The data is extracted from the selected studies by both the authors discretely in excel sheets according to the structure depicted in Appendix C. Later, the extorted data was assessed and some disagreements were discussed among authors. But still, the correctness of extraction by authors poses the internal validity threat.
External validity threat deals with incapability of deriving to the generalized conclusions of SLR results. The characteristics reported in section 2.3.4 tried to conclude maximum features from selected studies but there may be others which can be further investigated as the list is not exhaustive. The authors tried to summarize the findings of SLR from different aspects of estimation techniques but still, it might miss the in-depth analysis of the results.

- **Conclusion**

This study investigates the current state-of-art on test effort estimation in traditional software and mobile software/ application in traditional software development process and agile software development by means of Systematic Literature Review (SLR). During SLR, 75 studies are selected, searched from nine online data sources to answer four Research Questions (RQs). The main findings of the survey for RQ1 resulted in providing 26 test estimation techniques for traditional software centered on model-based approach, hybrid approach, *meta*-heuristic approach, analogy-based approach, and non-model based approach. Use Case Point and its extension and optimizations are prominently used in traditional software test estimation. But for the mobile application domain, a COSMIC method for function size estimation is prevailing in the literature survey. For estimation techniques reported in section 2.3.3.2 for agile mobile application development and testing, sturdy conclusions cannot be drawn due to lack of endemic studies in the literature. But results from section 2.3.3.1 on COSMIC FSM method for mobile applications can be further explored in an agile environment based on discussions presented by the Kamal Ramasubramani

(2016) that COSMIC FSM can be investigated for estimating testing efforts in agile projects. A comparison of test estimation techniques for traditional software and mobile application software is presented based on SLR. Later, mobile application characteristics are identified in SLR. It was comprehended that the mobile application characteristics are not enclosed by the present estimation techniques. There is no formal model that exclusively considers mobile application development and testing different from other traditional applications. Lastly, discussions and some research gaps are reported and certain future research avenues in the estimation of mobile apps in agile and traditional development and testing process are discussed.

Conducting the review phase

- Search strategy

The intent of the search strategy is to discover the studies that would assist in answering the RQs. The three phases of the search strategy comprise of identifying keywords and defining search strings, data sources selection and finally search process in data sources.

- Identifying keywords and defining search strings

The foremost phase of the search strategy is to ascertain the search string. The search strategy is set up to describe search strings and primary data sources. The guidelines provided by Kitchenham and Charters (2007) were followed to define the search string by analyzing the main keywords in RQs, synonyms of the keywords and on any other spellings of the words. Following are the identified keywords and synonyms are shown in Table 2:

List of keywords and synonyms.

Software, Software, project, system, application

Testing, Test, verification, validation

Effort, cost, resource, size, metric,

Estimation, Estimating, estimate, prediction, predicting, predict, assessment, forecasting, forecast, calculation, calculate, calculating, sizing, measure, measuring

Mobile Application, Mobile software, Mobile Apps, Mobile project

Development, Improvement, Progress

Method, Process, techniques, models, approaches

Agile, Scrum, XP, lean, crystal

Characteristics, Features, attribute, factors

Based on the identified keywords, the search string was obtained by joining synonymous terms using the 'OR', other keywords using logical 'AND' and wildcard character (*). Here wildcard character represents 0, 1, or any number of alphanumeric characters. The search string is categorized in four ways according to the RQs formed. Table 3 lists the categories and corresponding search string.

- Data sources

The digital databases that were used to search the keywords are SpringerLink, IEEE Xplore, ACM Digital Library, Elsevier Science Direct, Research Gate, CiteSeer, and InderScience.

- Search process in data sources

The next phase is to apply the search string to the chosen electronic data sources to find all the entailed studies. This phase is divided into two sub-phases: primary and secondary search phase. In the Primary Search Phase, the electronic data sources identified are examined based on the search string defined earlier. Initially, a total of 2467 results was retrieved with the chosen search string. These results from data sources are monitored to include search string in title and abstracts. The search string is again refined each time to check the outcome and analyzed for better results. Additionally, results are restricted to peer-reviewed conference papers and journal papers. The duplicate titles and abstracts are removed. In the secondary search phase, a technique called snowball tracking is used for studying all the references of primary studies to exploit further studies and increase the chances of inclusion of important papers in the systematic literature review. Table 4 lists the refined results from data sources after primary and secondary search phase.

SYSTEM DESIGN

- ## USE CASE DIAGRAM

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

Scenarios in which your system or application interacts with people, organizations, or external systems

Goals that your system or application helps those entities (known as actors) achieve

The scope of your system
A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

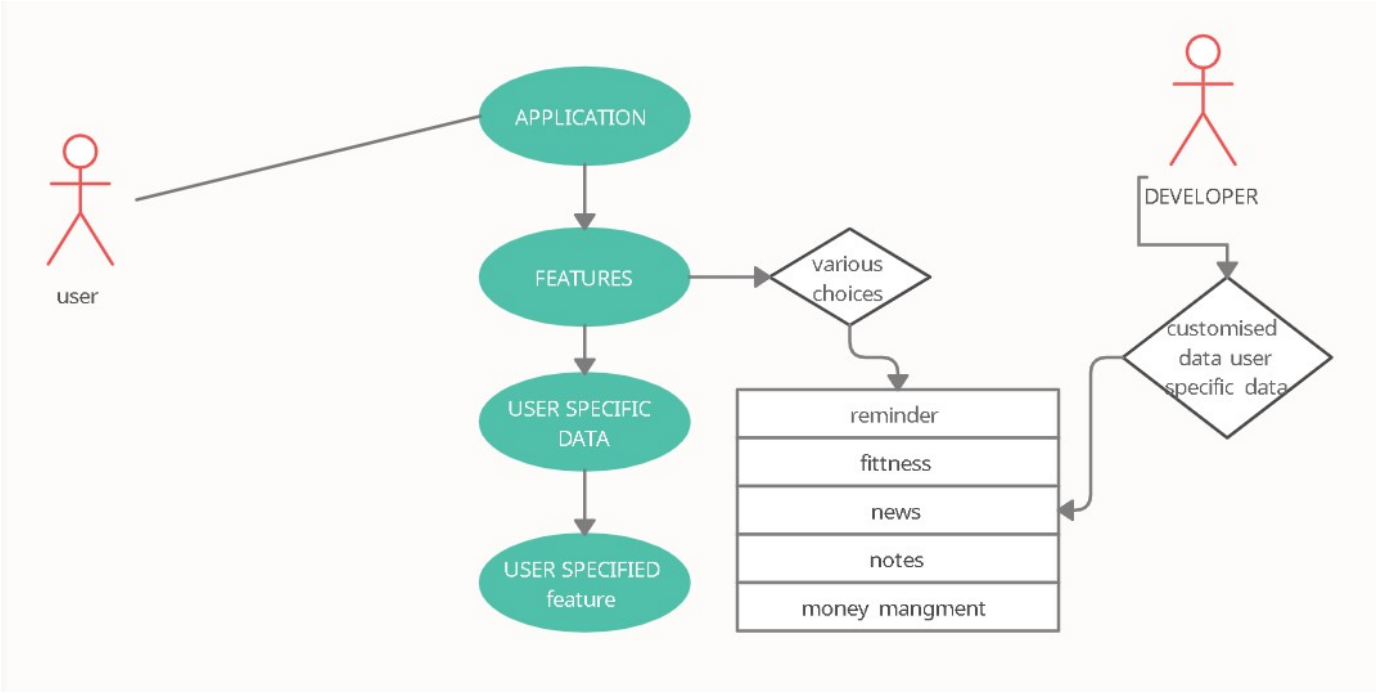

Fig. 3. Usecase diagram

- ## ACTIVITY DIAGAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as −

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

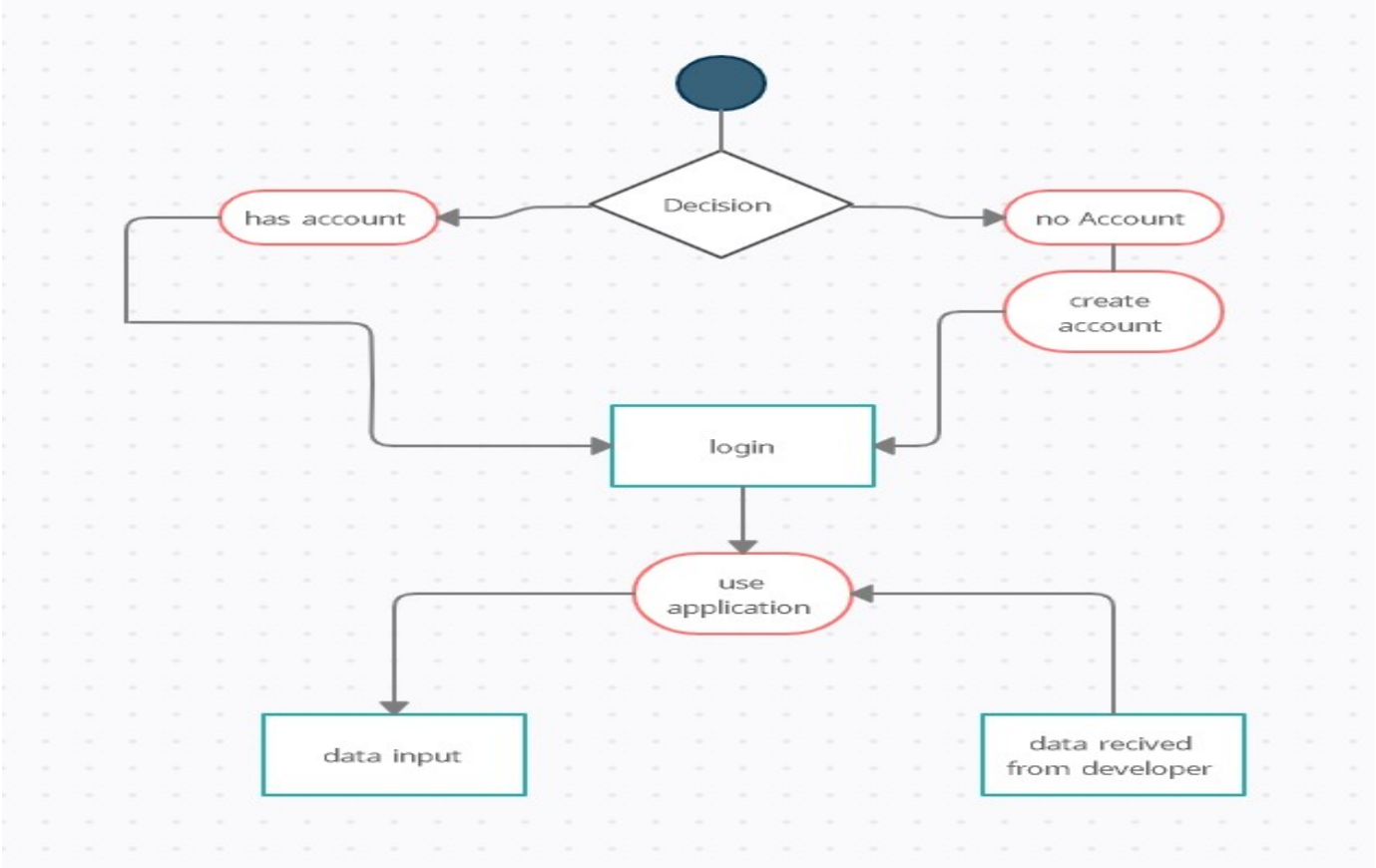

Fig. 4. Activity diagram

- **REQUIREMENT ANALYSIS**

• FUNCTIONAL REQUIREMENT

System must fulfil the given requirements

1. Must be able record new user
2. Must keep catalogue of all the services
3. Must record user review
4. Must record daily updates by user
5. Must provide daily reminders of fitness and weather

• NON-FUNCTIONAL REQUIREMENT

System must fulfil the following non-functional requirement on availability of internet

1. Availability 24*7
2. Customer data confidentiality
3. Better design and performance
4. Flexible service-based architecture

FUNCTIONALITY

- **API**

Application programming interfaces, or APIs, simplify software development and innovation by enabling applications to exchange data and functionality easily and securely.

What is an application programming interface (API)?

An application programming interface, or API, enables companies to open up their applications' data and functionality to external third-party developers, business partners, and internal departments within their companies. This allows services and products to communicate with each other and leverage each other's data and functionality through a documented interface. Developers don't need to know how an API is implemented; they simply use the interface to communicate with other products and services. API use has surged over the past decade, to the degree that many of the most popular web applications today would not be possible without APIs.

- How an API works

An API is a set of defined rules that explain how computers or applications communicate with one another. APIs sit between an application and the web server, acting as an intermediary layer that processes data transfer between systems.

Here's how an API works:

A client application initiates an API call to retrieve information—also known as a request. This request is processed from an application to the web server via the API's Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body.

After receiving a valid request, the API makes a call to the external program or web server.

The server sends a response to the API with the requested information.

The API transfers the data to the initial requesting application.

While the data transfer will differ depending on the web service being used, this process of requests and response all happens through an API. Whereas a user interface is designed for use by humans, APIs are designed for use by a computer or application.

APIs offer security by design because their position as middleman facilitates the abstraction of functionality between two systems—the API endpoint decouples the consuming application from the infrastructure providing the service. API calls usually include authorization credentials to reduce the risk of attacks on the server, and an API gateway can limit access to minimize security threats. Also, during the exchange, HTTP headers, cookies, or query string parameters provide additional security layers to the data.

For example, consider an API offered by a payment processing service. Customers can enter their card details on the frontend of an application for an ecommerce store. The payment processor doesn't require access to the user's bank account; the API creates a unique token for this transaction and includes it in the API call to the server. This ensures a higher level of security against potential hacking threats.

- Why we need APIs

Whether you're managing existing tools or designing new ones, you can use an application programming interface to simplify the process. Some of the main benefits of APIs include the following:

Improved collaboration: The average enterprise uses almost 1,200 cloud applications (link resides outside of IBM), many of which are disconnected. APIs enable integration so that these platforms and apps can seamlessly communicate with one another. Through this integration, companies can automate workflows and improve workplace collaboration. Without APIs, many enterprises would lack connectivity and would suffer from informational silos that compromise productivity and performance.

Easier innovation: APIs offer flexibility, allowing companies to make connections with new business partners, offer new services to their existing market, and, ultimately, access new markets that can generate massive returns and drive digital transformation. For example, the company Stripe began as an API with just seven lines of code. The company has since partnered with many of the biggest enterprises in the world, diversified to offer loans and corporate cards, and was recently valued at USD 36 billion (link resides outside of IBM).

Data monetization: Many companies choose to offer APIs for free, at least initially, so that they can build an audience of developers around their brand and forge relationships with potential business partners. However, if the API grants access to valuable digital assets, you can monetize it by selling access (this is referred to as the API economy). When AccuWeather (link resides outside of IBM) launched its self-service developer portal to sell a wide range of API packages, it took just 10 months to attract 24,000 developers, selling 11,000 API keys and building a thriving community in the process.

Added security: As noted above, APIs create an added layer of protection between your data and a server. Developers can further strengthen API security by using tokens, signatures, and Transport Layer Security (TLS) encryption; by implementing API gateways to manage and authenticate traffic; and by practicing effective API management.

Common API examples

Because APIs allow companies to open up access to their resources while maintaining security and control, they have become a valuable aspect of modern business. Here are some popular examples of application programming interfaces you may encounter:

Universal logins: A popular API example is the function that enables people to log in to websites by using their Facebook, Twitter, or Google profile login details. This convenient feature allows any website to leverage an API from one of the more popular services to quickly authenticate the user, saving them the time and hassle of setting up a new profile for every website service or new membership.

Third-party payment processing: For example, the now-ubiquitous "Pay with PayPal" function you see on ecommerce websites works through an API. This allows people to pay for products online without exposing any sensitive data or granting access to unauthorized individuals.

Travel booking comparisons: Travel booking sites aggregate thousands of flights, showcasing the cheapest options for every date and destination. This service is made possible through APIs that provide application users with access to the latest information about availability from hotels and airlines. With an autonomous exchange of data and requests, APIs dramatically reduce the time and effort involved in checking for available flights or accommodation.

Google Maps: One of the most common examples of a good API is the Google Maps service. In addition

to the core APIs that display static or interactive maps, the app utilizes other APIs and features to provide users with directions or points of interest. Through geolocation and multiple data layers, you can communicate with the Maps API when plotting travel routes or tracking items on the move, such as a delivery vehicle.

Twitter: Each Tweet contains descriptive core attributes, including an author, a unique ID, a message, a timestamp when it was posted, and geolocation metadata. Twitter makes public Tweets and replies available to developers and allows developers to post Tweets via the company's API.

Types of APIs and Types of API protocols

Types of APIs

Nowadays, most application programming interfaces are web APIs that expose an application's data and functionality over the internet. Here are the four main types of web API:

Open APIs are open source application programming interfaces you can access with the HTTP protocol. Also known as public APIs, they have defined API endpoints and request and response formats.

Partner APIs are application programming interfaces exposed to or by strategic business partners. Typically, developers can access these APIs in self-service mode through a public API developer portal. Still, they will need to complete an onboarding process and get login credentials to access partner APIs.

Internal APIs are application programming interfaces that remain hidden from external users. These private APIs aren't available for users outside of the company and are instead intended to improve productivity and communication across different internal development teams.

Composite APIs combine multiple data or service APIs. These services allow developers to access several endpoints in a single call. Composite APIs are useful in microservices architecture where performing a single task may require information from several sources.

Types of API protocols

As the use of web APIs has increased, certain protocols have been developed to provide users with a set of defined rules that specifies the accepted data types and commands. In effect, these API protocols facilitate standardized information exchange:

SOAP (Simple Object Access Protocol) is an API protocol built with XML, enabling users to send and receive data through SMTP and HTTP. With SOAP APIs, it is easier to share information between apps or software components that are running in different environments or written in different languages.

XML-RPC is a protocol that relies on a specific format of XML to transfer data, whereas SOAP uses a proprietary XML format. XML-RPC is older than SOAP, but much simpler, and relatively lightweight in that it uses minimum bandwidth.

JSON-RPC is a protocol similar to XML-RPC, as they are both remote procedure calls (RPCs), but this one uses JSON instead of XML format to transfer data. Both protocols are simple. While calls may contain multiple parameters, they only expect one result.

REST (Representational State Transfer) is a set of web API architecture principles, which means there are no official standards (unlike those with a protocol). To be a REST API (also known as a RESTful API), the interface must adhere to certain architectural constraints. It’s possible to build RESTful APIs with SOAP protocols, but the two standards are usually viewed as competing specifications.

APIs, web services, and microservices

A web service is a software component that can be accessed via a web address. Therefore, by definition, web services require a network. As a web service exposes an application's data and functionality, in effect, every web service is an API. However, not every API is a web service.

Traditionally, API referred to an interface connected to an application that may have been created with any of the low-level programming languages, such as Javascript. The modern API adheres to REST principles and the JSON format and is typically built for HTTP, resulting in developer-friendly interfaces that are easily accessible and widely understood by applications written in Java, Ruby, Python, and many other languages.

When using APIs, there are two common architectural approaches—service-oriented architecture (SOA) and microservices architecture.

SOA is a software design style where the features are split up and made available as separate services within a network. Typically, SOA is implemented with web services, making the functional building blocks accessible through standard communication protocols. Developers can build these services from scratch, but they usually create them by exposing functions from legacy systems as service interfaces.

Microservices architecture is an alternative architectural style that divides an application into smaller, independent components. Applying the application as a collection of separate services makes it easier to test, maintain, and scale. This methodology has risen to prominence throughout the cloud computing age, enabling developers to work on one component independent of the others.

While SOA was a vital evolutionary step in application development, microservices architecture is built to scale, providing developers and enterprises with the agility and flexiblity they need to create, modify, test, and deploy applications at a granular level, with shorter iteration cycles and more efficient use of cloud computing resources.

For a deeper dive on how these architectural approaches relate, see "SOA vs. microservices: What's the difference?"

- APIs and cloud architecture

It's crucial to develop APIs fit for purpose in today's world. Cloud native application development relies on connecting a microservices application architecture through your APIs to share data with external users, such as your customers.

The services within microservices architecture utilize a common messaging framework, similar to RESTful APIs, facilitating open communication on an operating system without friction caused by additional integration layers or data conversion transactions. Furthermore, you can drop, replace, or enhance any service or feature without any impact on the other services. This lightweight dynamic improves cloud resources optimization, paving the way for better API testing, performance and scalability.

- APIs and IBM Cloud®

APIs will continue to be just one part of application modernization and transforming your organization as the demand for better customer experiences and more applications impacts business and IT operations.

When it comes to meeting such demands, a move toward greater automation will help. Ideally, it would start with small, measurably successful projects, which you can then scale and optimize for other processes and in other parts of your organization.

Working with IBM, you'll have access to AI-powered automation capabilities, including prebuilt workflows, to help accelerate innovation by making every process more intelligent.

- **Data-base**

  - Database defined

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.
Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data.

  - What is Structured Query Language

SQL is a programming language used by nearly all relational databases to query, manipulate, and define data, and to provide access control. SQL was first developed at IBM in the 1970s with Oracle as a major contributor, which led to implementation of the SQL ANSI standard, SQL has spurred many extensions from companies such as IBM, Oracle, and Microsoft. Although SQL is still widely used today, new programming languages are beginning to appear.

  - Evolution of the database

Databases have evolved dramatically since their inception in the early 1960s. Navigational databases such as the hierarchical database (which relied on a tree-like model and allowed only a one-to-many relationship), and the network database (a more flexible model that allowed multiple relationships), were the original systems used to store and manipulate data. Although simple, these early systems were inflexible. In the 1980s, relational databases became popular, followed by object-oriented databases in the 1990s. More recently, NoSQL databases came about as a response to the growth of the internet and the need for faster speed and processing of unstructured data. Today, cloud databases and self-driving databases are breaking new ground when it comes to how data is collected, stored, managed, and utilized.

  - What's the difference between a database and a spreadsheet

Databases and spreadsheets (such as Microsoft Excel) are both convenient ways to store information. The primary differences between the two are:

- How the data is stored and manipulated

- Who can access the data
- How much data can be stored
  Spreadsheets were originally designed for one user, and their characteristics reflect that. They're great for a single user or small number of users who don't need to do a lot of incredibly complicated data manipulation. Databases, on the other hand, are designed to hold much larger collections of organized information—massive amounts, sometimes. Databases allow multiple users at the same time to quickly and securely access and query the data using highly complex logic and language.

    - **Types of databases**

  There are many different types of databases. The best database for a specific organization depends on how the organization intends to use the data.

  Relational databases

- Relational databases became dominant in the 1980s. Items in a relational database are organized as a set of tables with columns and rows. Relational database technology provides the most efficient and flexible way to access structured information.

  Object-oriented databases

- Information in an object-oriented database is represented in the form of objects, as in object-oriented programming.

  Distributed databases

- A distributed database consists of two or more files located in different sites. The database may be stored on multiple computers, located in the same physical location, or scattered over different networks.

  Data warehouses

- A central repository for data, a data warehouse is a type of database specifically designed for fast query and analysis.

  NoSQL databases

- A NoSQL, or nonrelational database, allows unstructured and semistructured data to be stored and manipulated (in contrast to a relational database, which defines how all data inserted into the database must be composed). NoSQL databases grew popular as web applications became more common and more complex.

  Graph databases

- A graph database stores data in terms of entities and the relationships between entities.
- OLTP databases. An OLTP database is a speedy, analytic database designed for large numbers of transactions performed by multiple users.
  These are only a few of the several dozen types of databases in use today. Other, less common databases are tailored to very specific scientific, financial, or other functions. In addition to the different database types, changes in technology development approaches and dramatic advances such as the cloud and automation are propelling databases in entirely new directions. Some of the latest databases include

Open source databases

- An open source database system is one whose source code is open source; such databases could be SQL or NoSQL databases.

Cloud databases

- A cloud database is a collection of data, either structured or unstructured, that resides on a private, public, or hybrid cloud computing platform. There are two types of cloud database models: traditional and database as a service (DBaaS). With DBaaS, administrative tasks and maintenance are performed by a service provider.

Multimodel database

- Multimodel databases combine different types of database models into a single, integrated back end. This means they can accommodate various data types.

Document/JSON database

- Designed for storing, retrieving, and managing document-oriented information, document databases are a modern way to store data in JSON format rather than rows and columns.

Self-driving databases

- The newest and most groundbreaking type of database, self-driving databases (also known as autonomous databases) are cloud-based and use machine learning to automate database tuning, security, backups, updates, and other routine management tasks traditionally performed by database administrators.

    - **Cloud Database**

A cloud database is a database service built and accessed through a cloud platform. It serves many of the same functions as a traditional database, with the added flexibility of cloud computing. Users install software on a cloud infrastructure to implement the database.

Key features

- A database service built and accessed through a cloud platform
- Enables enterprise users to host databases without buying dedicated hardware
- Can be managed by the user or offered as a service and managed by a provider
- Can support SQL (including MySQL) or NoSQL databases
- Accessed through a web interface or vendor-provided API

Benefits

Ease of access: Users can access cloud databases from virtually anywhere using a vendor's API or web interface.

Scalability: Cloud databases can expand their storage capacities on run-time to accommodate changing needs. Organizations only pay for what they use.

Disaster recovery: In the event of a natural disaster, equipment failure or power outage, data is kept secure through backups on remote servers.

Considerations

Control options: Users can opt for a virtual machine image managed like a traditional database or a provider's database as a service (DBaaS).
Database technology: SQL databases are difficult to scale but very common. NoSQL databases scale more easily but do not work with some applications.
Security: Most cloud database providers encrypt data and provide other security measures; organizations should research their options.Maintenance: When using a virtual machine image, one should ensure that IT staffers can maintain the underlying infrastructure.

- **ANDROID APPLICATION**

Basic working of android application framework

Android is a vast software system and it has been developed in a modular manner by using layers for it's specific functionality on top of the Linux kernel to speed up development and still be open source.

Various layers of Android are shown in the image below

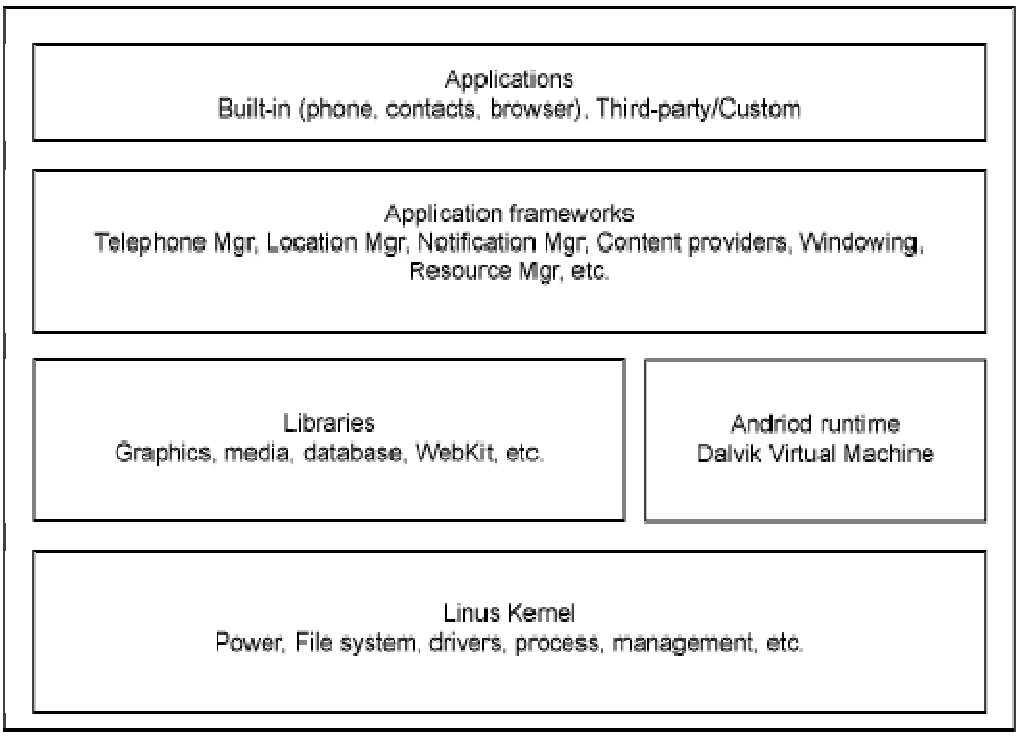

Fig. 5. Android layers

Linux Kernel – The lowest layer of the Android framework is the Linux kernel for low level hardware interfaces.

DVM / Libraries – Java programming language, is used for developing Android applications which are executed on a virtual machine (VM) instead of JVM and it is the Dalvik Virtual Machine (DVM). Every Android application runs in an instance of the DVM.

Applications – They are Android applications which are either custom developed or are provided with Android.

The layering is illustrated as

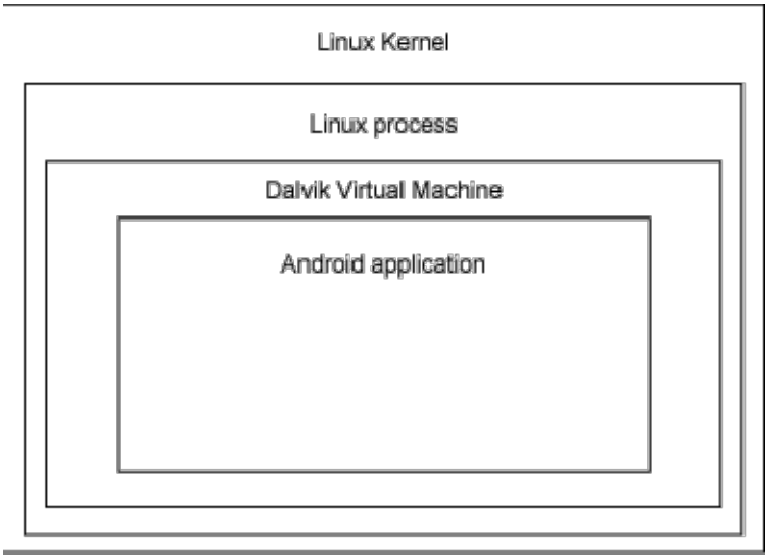

Fig. 6. layering

Android Application Framework – An Android application is actually a package of different components and each tasked for specific aim. It is not a executable as in case of Windows and has the following

Intents – It is usually an 'action' instruction and is a Java package name prefixed as 'action.COMMAND_NAME.'. IThey are actually events to which the application will reply to.

Services – They run in background and persist even though might not always be in top of screen.

BroadcastReceivers – They alert applications of events in the device.

Activity – It is the part of application with which user interacts with.

ContentProviders – They enable data sharing amongst Android applications

FUTURE SCOPE

The Scope of android app developers has increased and will increase in the coming years in India. Big companies like Samsung, Vivo, One Plus are launching mobile phones with a new update every year, that has increased the demand for android. There will be no loss in Android App Development in the coming years.

The **scope of android** app **development** is not limited to domains such as e-commerce, education, social media, gaming, banking & finance etc.

As android development and computer science is a ever developing field an android application with a flexible architectures can be ever evolving

CONCLUSION

This study investigates the current state-of-art on test effort estimation in traditional software and mobile software/ application in traditional software development process and agile software development by means of Systematic Literature Review It was comprehended that the mobile application characteristics are not enclosed by the present estimation techniques. There is no formal model that exclusively considers mobile application development and testing different from other traditional applications. Lastly, discussions and some research gaps are reported and certain future research avenues in the estimation of mobile apps in agile and traditional development and testing process are discussed.

The proper implementation of this project can help save people time and a lot of effort and as the scope of android development is infinite this project will forever be ever evolving.

REFERENCES

- Abdullah et al., 2014
  Abdullah, N.A.S., Rusli, N.I.A., Ibrahim, M.F., 2014. Mobile game size estimation: COSMIC FSM rules, UML mapping model and Unity3D game engine, in: ICOS 2014–2014 IEEE Conference on Open Systems. pp. 42–47.

- Abhilasha and Sharma, 2013
  Abhilasha, Sharma, A., 2013. Test effort estimation in regression testing, in: Innovation and Technology in Education (MITE), 2013 IEEE International Conference in MOOC. pp. 343–348.

- Abhishek et al., 2010 C. Abhishek, V.P. Kumar, H. Vitta, P.R. Srivastava
  Test effort estimation using neural network
  J. Softw. Eng. Appl., 03 (2010), pp. 331-340
- Abran et al., 2007 A. Abran, J. Garbajosa, L. Cheikhi
  Estimating the test volume and effort for testing and verification & validation
  IWSM-MENSURA Conference (2007), pp. 216-234
- Aloka et al., 2011S. Aloka, P. Singh, G. Rakshit, P.R. Srivastava
  Test effort estimation-particle swarm optimization based approach
  Commun. Comput. Inform. Sci. (2011), pp. 463-474
- Amalfitano et al., 2011
  Amalfitano, D., Fasolino, A.R., Tramontana, P., 2011. A GUI crawling-based technique for android mobile application testing, in: Proceedings – 4th IEEE International Conference on Software Testing, Verification, and Validation Workshops, ICSTW 2011. pp. 252–261.
- Aranha and Borba, 2007a
  Aranha, E., Borba, P., 2007a. Test Effort Estimation Models Based on Test Specifications, in: Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION, 2007. TAICPART-MUTATION 2007. pp. 1–5.
- Aranha and Borba, 2007 E. Aranha, P. Borba
  An Estimation Model for Test Execution Effort, in: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)
  IEEE (2007), pp. 107-116
- Aranha and Borba, 2007b
  Aranha, E., Borba, P., 2007b. Empirical studies of test execution effort estimation based on test characteristics and risk factors, in: Doctoral Symposium on Empirical Software Engineering (IDoESE 2007).
- Arumugam and Babu, 2015 C. Arumugam, C. Babu
  Test size estimation for object oriented software based on analysis model
  J. Softw., 10 (2015), pp. 713-729
- Arzenšek and Heričko, 2014 B. Arzenšek, M. Heričko
  Criteria for selecting mobile application testing tools
  CEUR Workshop Proceed. (2014), pp. 1-8
- Asghar et al., 2016
  M.Z. Asghar, A. Habib, A. Habib, S.R. Zahra, S. Ismail

  AndorEstimator: android based software cost estimation application

arXiv Prepr arXiv, 14 (2016), pp. 192-202

- Aslam et al., 2017
  W. Aslam, F. Ijaz, Muhammad Ikram Lali, Waqar Mehmood

  Risk aware and quality enriched effort estimation for mobile applications in distributed agile software development
  J. Inf. Sci. Eng., 33 (6) (2017), pp. 1481-1500
- Badri et al., 2015
  M. Badri, F. Toure, L. Lamontagne

  Predicting unit testing effort levels of classes: an exploratory study based on multinomial logistic regression modeling
  Proced. Comput. Sci. (2015), pp. 529-538
- Bhattacharya et al., 2012
  P. Bhattacharya, P.R. Srivastava, B. Prasad

  Software test effort estimation using particle swarm optimization
  Adv. Intell Soft Comput., 132 AISC (2012), pp. 827-835

- Bhattacharyya and Malgazhdarov, 2016
  Bhattacharyya, A., Malgazhdarov, T., 2016. PredSym: estimating software testing budget for a bug-free release. Proc. 7th Int. Work. Autom. Test Case Des. Sel. Eval. – A-TEST 2016 16–22.
- Cao, G., Yang, J., Zhou, Q., Chen, W., 2012. Software Testing Strategy for Mobile Phone, Advances and Applications in Mobile Computing. InTech.

- Catolino, G., Salza, P., Gravino, C., Ferrucci, F., 2017. A Set of Metrics for the Effort Estimation of Mobile Apps, in: Proceedings - 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2017. pp. 194–198.

- Charland, A., Leroux, B., 2011. mobile application Development : Web vs . native. Commun. ACM.
- M. Ciman, O. Gaggi
  An empirical analysis of energy consumption of cross-platform frameworks for mobile development
  Pervasive Mob. Comput., 39 (2017), pp. 214-230

- P. Costa, A.C.R. Paiva, M. Nabuco
  Pattern based GUI testing for mobile applications, in: Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology
  QUATIC (2014), pp. 66-74
- T.F.V.D. Cunha, V.L.L. Dantas, R.M.C. Andrade
  SLeSS: A scrum and lean six sigma integration approach for the development of sofware customization for mobile phones, in: Proceedings - 25th Brazilian Symposium on Software Engineering

SBES (2011), pp. 283-292

- L. D'Avanzo, F. Ferrucci, C. Gravino, P. Salza
  COSMIC functional measurement of mobile applications and code size estimation, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing
  - SAC '15 (2015), pp. 1631-1636
- Dalmasso, I., Datta, S.K., Bonnet, C., Nikaein, N., 2013. Survey, comparison and evaluation of cross platform mobile application development tools, in: 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC). pp. 323–328.

- V.L.F.G.M. Dantas, A.L. da Costa, R.M.C. Andrade
  Testing requirements for mobile applications, in: 2009 24th International Symposium on Computer and Information Sciences
  ISCIS (2009), pp. 555-56
- G. de Cleva Farto, A.T. Endo
  . Reuse of model-based tests in mobile apps, in: Proceedings of the 31st Brazilian Symposium on Software Engineering - SBES'17
  ACM Press, New York, New York, USA (2017)
  pp. 184–193
- de Souza and Aquino, 2014
  L.S. de Souza, G.S. Aquino Jr, de

  MEFFORTMOB: A Effort Size Measurement for Mobile Application Development
  Int. J. Softw. Eng. Appl., 5 (2014), pp. 63-81
- Dewi and Nur Atiqah Sia, 2015
  Dewi, M.., Nur Atiqah Sia, A., 2015. Reviews on agile methods in mobile application development process, in: 2015 9th Malaysian Software Engineering Conference (MySEC). pp. 161–165.
- Farooq et al., 2011
  S.U. Farooq, S.M.K. Quadri, N. Ahmad

  Software measurements and metrics: role in effective software testing
  Int. J. Eng. Sci. Technol., 3 (2011), pp. 671-680
- Ferrucci et al., 2015
  Ferrucci, F., Gravino, C., Salza, P., Sarro, F., 2015. Investigating Functional and Code Size Measures for Mobile Applications, in: Proceedings - 41st Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2015. pp. 271–287.
- Francese et al., 2015
  Francese, R., Gravino, C., Risi, M., Scanniello, G., Tortora, G., 2015. On the Use of Requirements Measures to Predict Software Project and Product Measures in the Context of Android Mobile Apps: A Preliminary Study, in: Proceedings - 41st Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2015. pp. 357–364.
- Franke et al., 2012
  Franke, D., Kowalewski, S., Weise, C., Prakobkosol, N., 2012. Testing conformance of life cycle dependent properties of mobile applications, in: Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012. pp. 241 – 250.

- Gao et al., 2014
  Gao, J., Bai, X., Tsai, W.-T., Uehara, T., 2014. Mobile Application Testing: A Tutorial. Computer (Long. Beach. Calif). 2, pp. 46–55.
- Giessmann et al., 2012
  Giessmann, A., Stanoevska-Slabeva, K., de Visser, B., 2012. Mobile Enterprise Applications – Current State and Future Directions, in: 45th Hawaii International Conference on System Sciences. pp. 1363–1372.
- Göth, 2015
  B.R. Göth

  Testing Techniques for Mobile Device Applications
  Masaryk University (2015)
- Haoues et al., 2017
  Haoues, M., Sellami, A., Ben-Abdallah, H., 2017. A Rapid Measurement Procedure for Sizing Web and Mobile Applications Based on COSMIC FSM Method, in: Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement. pp. 129–137.
- Hauptmann et al., 2014
  Hauptmann, B., Junker, M., Eder, S., Amann, C., Vaas, R., 2014. An expert-based cost estimation model for system test execution, in: ACM International Conference Proceeding Series. pp. 159–163.
- Heeringen and Gorp, 2014
  Heeringen, H. Van, Gorp, E. Van, 2014. Measure the functional size of a mobile app: Using the cosmic functional size measurement method, in: Proceedings - 2014 Joint Conference of the International Workshop on Software Measurement, IWSM 2014 and the International Conference on Software Process and Product Measurement, Mensura 2014. pp. 11–16.
- Holl and Elberzhager, 2016
  Holl, K., Elberzhager, F., 2016. Quality Assurance of Mobile Applications : A Systematic Mapping Study, in: 15th International Conference on Mobile and Ubiquitous Multimedia ACM, New York, NY, USA,. pp. 101–113. https://doi.org/10.1145/3012709.3012718
- Islam et al., 2016
  Islam, S., Pathik, B.B., Khan, M.H., Habib, M., 2016. Software test estimation tool: Comparable with COCOMOII model, in: IEEE International Conference on Industrial Engineering and Engineering Management. pp. 204–208.
- Jayakumar and Abran, 2013
  K.R. Jayakumar, A. Abran

  A survey of software test estimation techniques
  J. Softw. Eng. Appl., 6 (2013), pp. 47-52
- Jin and Jin, 2016
  C. Jin, S.W. Jin

Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization
Appl. Soft Comput. J., 40 (2016), pp. 283-291

- Kamala Ramasubramani, 2016
  Kamala Ramasubramani, J., 2016. ESTIMATION MODEL FOR SOFTWARE TESTING. ÉCOLE DE TECHNOLOGIE SUPÉRIEURE UNIVERSITÉ DU QUÉBEC.
- Katherine and Alagarsamy, 2012
  a.V. Katherine, D.K. Alagarsamy

Conventional software testing vs cloud testing
Int. J. Sci., 3 (2012), pp. 1-5

- Kaur, 2016
  A. Kaur

Review on agile approach to mobile application development
IJCAT – International J. Comput. Technol., 3 (2016), pp. 200-203

- Kim et al., 2009
  Kim, H., Choi, B., Wong, W.E., 2009. Performance testing of mobile applications at the unit test level, in: SSIRI 2009 – 3rd IEEE International Conference on Secure Software Integration Reliability Improvement. pp. 171–181.

- Kim, H.K., 2012. Mobile applications software testing methodology, Computer Applications for Web, Human Computer Interaction, Signal and Image Processing, and Pattern Recognition. Communications in Computer and Information Science. Springer, Berlin, Heidelberg.

- Kirubakaran, B., Karthikeyani, V., 2013. Mobile application testing — Challenges and solution approach through automation, in: 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering. pp. 79–84.

- Kitchenham, B., Charters, S., 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering.

- D.S. Kushwaha, A.K. Misra

Software test effort estimation
ACM SIGSOFT Softw. Eng. Notes, 33 (2008), pp. 1-6

- Lazić, L., Mastorakis, N., 2009. The COTECOMO: COnstractive Test Effort COst MOdel, in: European Computing Conference. Lecture Notes in Electrical Engineering. Springer, Boston, MA.