Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Naeem Ahmad Sattar

SELECTION OF LOW-CODE PLATFORMS BASED ON ORGANIZATION AND APPLICATION TYPE

Examiners: Paula Savolainen
Professor Ajantha Dahanayake

Supervisor: Paula Savolainen

ABSTRACT

Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Naeem Ahmad Sattar

Selection of Low-Code Platforms Based on Organization and Application Type

Master's Thesis

86 pages, 7 figures, 30 tables

Examiners: Paula Savolainen

Professor Ajantha Dahanayake

Keywords: Low-Code Platform, Assessment Criteria, Programming Languages

The demand of applications is increasing rapidly, however, world is not producing the developers in that ratio. Moreover, the development companies are facing numerous challenges to meet the requirements of the clients, who demand the applications with no time. To overcome this gap of developers, there are numerous Low-Code Platforms available who not only enables the developers (technical employees) to develop the applications in a faster way, but also enables the citizen developers (non-technical employees) to develop the applications. Low-Code Platforms provides built in tools to develop the applications with minimum code or without code.

Now the problem is to find out which Low-Code Platform will be selected for which type of organization and which type of application. The thesis provides the decision tree to select the suitable Low-Code Platform based on organization and application type.

Moreover, the thesis also concentrates on assessment criteria for the Low-Code Platforms to show the standings of Low-Code Platform vendors.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Paula Savolainen for giving me freedom to work independently and providing me innovative topic for the thesis work. Her guidance kept me on the right track. I am gratefully indebted to her for her very valuable comments on this thesis.

I would also like to express my deep appreciation and respect to Prof. Ajantha Dahanayake for her support.

I would also like to thank “Forrester” for providing me couple of reports as the academic contribution.

I would also like to thank my mother for everthing she did for me.

Last but not the least, a highly special thanks to my dear wife for pushing me to complete my thesis in time.

Special thanks to my sweet children.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

1GL	First Generation Languages
2GL	Second Generation Programming Languages
3GL	Third Generation Programming Languages
4GL	Fourth Generation Programming Languages
AD&D	Application Design & Development
CPU	Central Processing Unit
CRUD	Create Read Update Delete
DPA	Digital Process Automation
ENIAC	Electronic Numerical Integrator and Computer
IOS	Internetwork Operating System
LPL	Logical Programming Languages
PAAS	Platform as a Service
PL	Programming Languages
UI	User Interface
UX	User Experience

1 INTRODUCTION

1.1 Background

Applications development demand is increasing rapidly, however, the world is not producing IT developers with that pace. Till 2021, the application development demand in the market will be at least five times than the development capacity to meet the demand [1]. To meet the shortfall of IT developers we need alternatives that no only help us to create applications in a quick way, also enables the citizen developers to develop their own applications. In the current era, we have solution of the problem which is Low-Code Platforms that enables the IT developers to create the applications quickly. Moreover, these platforms enable the citizen developers to create their own applications. Low-Code application development term is new and few years ago no one has the idea about Low-Code [6].

Low-Code is an approach where applications design and develop is performed in a Digital Speed with less coding. This approach provides the opportunity to the skilled developers to develop more reliable applications with fast speed. It enables the developers to skip all basic steps which can stall them so that they can straight forward get the 10 % unique of the application [7].

1.1.1 What is Platform?

To understand Low-Code platform, first we need to look into couple of definitions of platform.

"A platform is a group of technologies that are used as a base upon which other applications, processes or technologies are developed" [2].
"In IT, a platform is any hardware or software used to host an application or service" [3].

1.1.2 Why Low-Code Platform?

In the current era, Business and IT works together. For any application and business, time to market is a critical issue. Businessmen almost always pushing the technical teams to provide their business solutions with no time. To take the advantage of time to market,

business and IT should work on Digital Speed. Low-Code can be the option to meet up the Digital Speed development.

1.1.3 What is Low-Code Platform?

"Platforms that enable rapid delivery of business applications with minimum hand-coding and minimal upfront investment in setup, training, and deployment" [4].

"For a growing number of business concerns, Low-Code is the way forward. It's a streamlined development process faster than any that's come before it. Businesses need to change at a rate that can keep up with competitors, vendors, and the modern consumer's fleeting stream of impatient desires" [5].

Low-Code development / programming platforms are mostly used in sense of developing the applications with minimal human effort required which caters of the view that applications that are developed in a rapid method. The Low-Code Development Platforms (LCDP) are based on graphical user interface in designing the application as opposed to the hard-coded programming techniques. The feature of LCDP focuses in the development of the Databases, Business processes, and User interface [8]

Low-Code Programming technique is basically derived from fourth generation programming ideology along with the concepts of RAD kept in sight. The Low-Code Programming enables the programmer to think less on the syntax of the code and put more emphasis in designing the esthetics and functionality of the application allowing lesser time required on troubleshooting and implementing. allowing to develop entirely operational applications [9].

Low-Code is generally referred to as novel and much new concept in the field programming with the idea being pitched in 2011 alone. The concept has changed into reality, but it has also come with a cost of criticism by the programmer community which believe that such platforms will reduce the functional concepts of programming and the field will no longer be suitable for professionals. Model driven software development approach, rapid application development, automatic code generation and visual

programming are the approaches that became the foundation of Low-Code Programming [10]. All these techniques have all been merged together to achieve the Low-Code Programming technology that is in front of us. Mendix, Salesforce, Appian, and PowerApps are some of the Low-Code Programming Languages.

Globally the Low-Code Programming is getting famous and lot of new developers have found their interest in this programming technique. The reason for this evolution in programming is that the tedious task of writing down the syntax and then troubleshooting it was more laborious and time consuming. Ultimately the ideas that were in mind were mostly bounded by the coding techniques. Therefore, stopping the functionality of the apps being developed to a certain bound, but with the Low-Code Programming now in the horizon new avenues have been opened. Especially with IOS and Android becoming the major operating systems. The need of smartphone apps increasing day by day, we are going to see the advent of the Low-Code Programming finding its way into the niche market at a rapid pace as of today.

1.1.4 Low-Code Users

Low-Code users are divided into two categories i.e. Citizen Developers and IT Developers.

"A developer is an individual that builds and create software and applications. He or she writes, debugs and executes the source code of a software application [108]."

A developer is also known as a software developer, computer programmer, programmer, software coder or software engineer.

Citizen Developers: According to Gartner "A citizen developer is a user who creates new business applications for consumption by others using development and runtime environments sanctioned by corporate IT" [11].

1.2 Aims and Objectives

The aim and objective of this study is to provide vendors assessment criteria, and selection of the suitable Low-Code Platforms based on the application type and organization type.

1.3 Research Questions

Table 1. Research Questions

Research Questions	Expected Outcome
RQ1: What is the contribution of the literature in Low-Code Platforms assessment and selection?	**EO1:** Provide Contribution of literature in Low-Code Platform assessment and selection
RQ2: Which is the suitable Low-Code Platform for which application and organization type?	**EO2:** Provide the suitable Low-Code Platform based on application organization type
RQ3: Which Low-Code Platform's vendor stands at which place in the assessment?	**EO3:** Provide the standings of Low-Code Platform's vendor.

1.4 Structure of the thesis

The Section 2 contains the Literature Review. In Section 3, the discussion is about why the selection of Low-Code Platform is important, and assessment criteria for Low-Code Platforms. The Section 4 contains Assessment Criteria for Low-Code Platforms and Decision Tree for selection of Low-Code Platform based on organization and application types. Section 5 contains Conclusions.

2 LITERATURE REVIEW

Low-Code is a new development approach, so the relevant literature review is limited. However, Low-Code Platform literature review can be seen in context of how the Programming Languages were evolved. To have the better understanding of Low-Code development, we will go in deep the way Programming Languages were developed and how they were evolved.

The focus of this literature review is the understanding of how Low-Code Programming has its origin of inception and how the Low-Code Platforms has been in the process of evolution. This literature review is aimed at the evolution of Low-Code Programming Languages. What journey did the Programming Languages take to reach this milestone? We will establish the fact, that the weaknesses of one generation was resolved in the next generation and the fact that most of the languages had retained their existence due to the flexibility to accept the changes.

The evolution of mankind has been subject to changes and ease of doing task. From the beginning Homo Sapiens have sought to increase the efficiency of task with minimum effort required. The invention of wheel to steam engine are the example of easing up human effort as much as possible. Computer as machine is also designed by human race to minimize the computations which may take week if done manually. From Electronic Numerical Integrator and Computer (ENIAC) to present day computers all are designed and run by the computer programs which are tasked for doing the specific jobs [12]. Designing computer programs is a tedious and time-consuming job. With the present-day demand of providing projects within small deadline the burden has fallen greatly, on the programmers. The evolution of Programming Languages has evolved them to the level of Low-Code Programming. Low-Code Programming is as such designed to reduce the human effort by minimizing the stress of programming syntax and structures [9]. Moreover, with add drop mechanism the designer can develop programs which are more designed based, and less syntax based, with less errors and redundant the programmer is able to deliver project on time. The evolution of Programming Languages to the Low-Code Platforms and how this platform will define the future is going to be the agenda of this

literature review.

2.1 Programming Languages Background

Programming finds the roots when Electronic Numerical Integrator and Computer (ENIAC) was created, which was the first ever computer system. The machine code was written in the form of batch files to run the ENIAC. This was the foundation of Programming Languages. The machine code and machine (ENIAC) both were dependent on each other, as code is nothing without machine and machine is nothing without code.

The computer architecture is based on a Central Processing Unit (CPU). The main purpose of CPU is to execute instruction generated from programmed codes [13]. The basic languages are series of binary digits and they form the backbone of basic programming as the machine system only understands with the binary digits [13]. Through the years many different Programming Languages had been designed and implemented in the market, which were categorized in the form of the generations and through these generations we can traverse how the programming language has evolved to its present different forms especially the Low-Code Programming techniques.

The evolution of programming language can be distributed between generations with each generation having its own set of specific programming trends that dictated the overall development scheme of that generation. In a collective manner there are five generations till date which are categorized over their functional trends and overall processing virtues. This specific literature review shall be focusing upon these generations and how each generation set as precursor for the successive generations to come and how each generation with its specific Programming Language had an impact on society and innovation [14]. These will be the focus of the discussion in this literature review altogether.

The fact remains that the Low-Code Programming has evolved as the Programming Languages have evolved. Therefore, it is imperative that we discuss these generation as well. A tabular notation of the generation with the years of introduction and related programing languages are shown in Table 2 below.

Table 2. Programming Languages and the generations over the year [15]

Sr #	Generation	Year	Popular languages
1	First Generation (1GL)	Early 1950	Flow-Matic
2	Second Generation (2GL)	Early 1950	Assembly languages
3	Third Generation (3GL)	1964	COBOL, FORTRAN, LISP
4	Fourth Generation (4GL)	1970	Pearl, php, Python ruby
5	Fifth Generation (5GL)	1980	Prolog, OSPS5

Low-Code Programming is the eventual evolved form of the Programming Languages. The main focus of the Low-Code Programming was to make sure that ease of coding be there. The Low-Code Programming Language was designed to make the programmer focus on the logic rather than notations and semantics. Let the programing language take care of the things. However, still the presence of Low-Code Programming and its evolution in becoming a high-end application development language is one of the follow ups of this literature review.

2.2 Segmentation of Programming Languages

Languages which are designed artificially for the communication with computer or passing instruction to computer are known as Programming Languages. There are some programs in Programming Languages (PL) which control the behavior of the computer and express the algorithms which are being used in the instructions. Programming Languages are divided into two main categories which are Imperative PL and Declarative PL [16] [17].

Imperative PL is also known as procedural programming. Imperative PL as commonly known the program is constructed with the help of one or more procedures, these procedures can be functions or the subroutines. Object oriented programming encompasses the approach where structured programming has been used to promote for the improvement of the maintainability and quality of the imperative programs. In imperative programming, assignment statements are used for storage of information in memory to use

it later after some time when it is required. Using the looping statements allows sequences of statements execution at one time. The Important characteristics of Imperative PL are absorbing changes, common data structures, user friendliness and global variables. Because of these points, Imperative PL are always preferred to work. The Imperative Programming Languages includes FORTRON, C++, JAVA, MATLAB, PHP [18].
Declarative PL is opposite to the Imperative PL. This allows the programmer to declare what should be done. Development of this style implies discussion of the logic of data processing but not its control flow. Developers describe the required results without the explicit description of the steps required to application. In Dclarative PL there are further two subclasses which are

1) Logical Programming Language
2) Functional Programming Language

"Logic programming is a computer programming paradigm in which program statements express facts and rules about problems within a system of formal logic" [19]. Programming Languages classes and subclasses of the Declarative PL, that is based on the use of logical formulas. Logical Rules of the programs are written as clauses with a proper head and a complete body. PROLOG and LISP are the main Logical Programming Languages (LPL) which are categorized in the Logical Programming Language (LPL) with their different dialects as well. The dialects of LISP are EuLISP, NewLISP, ISLISP, Common LISP. Functional Programming Languages are particularly designed for symbolic computation handling and the list processing applications. That language is relying on mathematical functions some of the well-known Programming Languages in this category are Python, Haskell, Erlang. Functional Programming Languages are further categorized into two main sub groups these are 1) Pure Functional Languages 2) Impure Functional Languages [20].

- **Pure Functional Languages** – These functional languages support functional paradigms individually like Haskell.
- **Impure Functional Languages** – These functional languages support the functional paradigm and support imperative style programming as well like LISP

2.2.1 Notations

Notations are used for writing programs in computer language which are specified in computational levels and have the importance of being followed upon and being used as. Along with these, the Programming Language also hold the specifications and implementations of how these languages are being used as per the need [21].

2.2.2 Syntax

The reason of discussing the semantics and syntax was to make the reader appreciate the fact that these two main parameters were responsible in the defining the next generations of Programming Languages and the reason of how the advanced forms of application development Programming Languages were designs such as the Low-Code Programming Language.

Syntax of any language defines how it is being narrated and understood by the reader [19]. In the computer Programming Language dimensions, the syntax of Programming Language is usually defined by using a combination of consistent expressions (for lexical structure). For the understanding of the grammar, syntax practices are used and are helpful for using the grammar in the code. For instance, the syntax in assembly language will be different from what we see in the web-based Programming Language although the conventions of using the mathematical formulas and other related aspect shall be the same.

2.2.3 Semantics

Semantics are basically the program building blocks. Such blocks maybe the examples of operational algebraic, axiomatic, denotational. Specifications means to determine the behavior of the program and its level of accuracy according to its source code file. whereas Implementations means execution of a program on one or more than one configurations of software and hardware. For implementation two different approaches are used for the configuration of hardware and software which are Interpretation and Compilation. Any language can be implemented using one of the configuration.

2.3 Low-level coding vs High-level coding

Before moving on the later stages of paradigms of the Programming Languages, this part has a very important part to play in the understanding of this literature review as what is the difference between the low-level coding and high-level coding and why are these two terms used. The evolution of the Programming Language and their history had been influenced by the need of the literature in the language as well. Since the developers are humans, it was seen imperative that the languages that were being used to design the applications must also be written in the human understandable languages so that the troubleshooting can be made easy and understandable for the other developers to continue with the work.

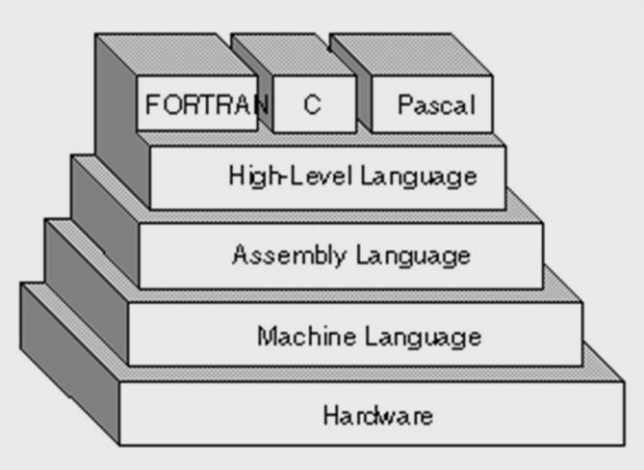

Fig. 1. An impression of the high-level vs low-level Programming Languages. [22]

The high-level programming is implemented for the code to be easier. Making the language understandable for the novice programmers. The high-level programming is portable as compared with the low-level programming. The reason of designing the high-level Programming Language was that the machines were becoming more versatile in the early 80's and the need for Programming Language was seen which were implementable on cross platform architectures was available with any machine running [23]. Since the designers were coming up with hardware's which used totally different sets of registers. The low-level programs were defined to only a particular system. So, the need was felt to design apps and operating system that could run on different platforms and perform the same tasks uniformly [24].

Comparatively, Low-Code Programming is platform dependent. This is the main disadvantage as the implementation on cross platform is not possible. Since they are used to write different driver signatures and ASICS also the firmware Languages are still being implemented using the low-level Languages. Secondly, an important factor that must be understood about the low-level code is that they are basically machine codes and thus, are very fast as the computations speeds are faster as no compiler is involved in between the kernel and the code [25]. After establishing the difference between the low-level and high-level programming, now let's proceed to the later topics of our literature review.

Table 3. Examples of Low and High-Level Programming Languages [20].

Sr #	Type of language	Example
1.	Low-level Programming Language	Assembly languages
2.	High-level Programming Languages	COBOL, FORTRAN, ALGOL, C, JAVA, PHP

2.4 Generations an overview

First ever Programming Language was developed by Alain Colmerauer when he designed Prolog [26]. Until today, almost five computer Programming Languages generations are introduced. To achieve the different demands of application programs, thousands of Programming Languages are introduced. These generations are based on the level of interaction with the computer hardware. Many languages are developed for research exploration and some of these were designed only for programming perspective. In scientific computation, FORTRAN was the first high-level language which were introduced at the time of low-level languages were being used. That is still known as the most influential Programming Languages in the history of Programming Languages.

2.4.1 First Generation Languages (1GL)

The first ever Programming Language existed well before the ancient history was recorded as the ancient Egyptians had used different mathematical programs to construct their architectures [16], the recorded program could be traced back into the past when

the main programming was placed only for some simple uses as we see them today. However, our scope of discussion is based on the existing Programming Languages and how they have evolved into what they are now as shown in the Table 2. The first ever computer program that was introduced was the FORTRAN that was designed by IBM and is considered as the oldest Programming Language.

The early Programming Language generation as the 1GL is considered to be low-level programming, before moving further into the details of the first-generation language let me first oversee what the low-level programming actually is and how it differs from others and what imperative does it follows. Basically, the low-level programming is defined as to be machine specific Programming Language, as we have understood that a computer system only understands binary language in the form of one's and zero's i.e. a code had to be written in such a way that the machine could understand it and then perform the tasks imbedded into the code that was written in the low-level programming technique. FORTRAN is multi-dimensional as it was first introduced it was simple Low-Code Platform [48]. With the applicability of FORTRAN being seen as more important, it has evolved as High-level Programming Language as we still see in later generations. FORTRAN is one of those low-level Programming Languages and ever since its inception in early 50's it is still used by many space and research-oriented facilities all over the world. One more thing about the low-level programming was its level of complexity for human understanding as the syntax of the language was hard to even understand [25].

Fig. 2. A set of vacuum tubes that were used in the first ever designed computer systems and the program were hard coded through these vacuum tubes [27] [28]

The above figure shows the first ever vacuum tubes that were used in the computer machines and their main task was to run the code in the form of binary numbers, basically the programmer had to physically code each of the vacuum tube to form the basis of the main computer program [27].

The main attractions of the first generations of Programming Language was the absence of any kind of assembler or compiler which would turn the written code into machine level syntax [29] thus the first generation of Programming Language was all but very complex of binary instructions that were hard to learn and implement.

2.4.2 Second Generation Programming Languages (2GL)

The second generation of Programming Language also referred as the assembly language. It had the same origins as the first generation of the Programming Language had in the sense that the both the languages were based on low-level programing platform. However, the real difference that enhanced the second generation language from the first generation languages was the provision of the assembler or the compiler. The assembler or compiler which converted the program into machine level understandable language that was zeroes and ones [30]. The assembler helped in making the programming the code much easier as it used the mnemonics codes for interpretation of the code. These codes had a single word instruction that could be used to send in a command to the system. Thus, providing an ease to the developer whose major overhead was resolved using the second generation Programming Language assembler. Features of assembler also came with its own assembler thus, becoming a second-generation Programming Language [31].

2.4.3 Third Generation Programming Languages (3GL)

As we have earlier established the difference between the high and low-level Programming Languages and why the need was felt for high-level language. The needs brought the situation which moved us to the third generation of Programming Languages 3GL which is based solely on the high-level language narratives.

As a preliminary introduction, the third-generation Programming Language was introduced in 1964-1971. The third generations languages are termed as high-level Programming

Languages, as the languages in the cadre are more machine independent and user friendly [32]. The new and advanced generation of Programming Language made the programming as platform independent meaning that code content was not dependent on the hardware which was being used.

Another more important feature of the third generation of Programming Languages was its ability to form the higher level of abstraction that the third generation of Programming Languages had to offer [33]. The level of abstraction provides the code to be more redundant to the network related threats since the hardware and system networks had evolved in the same era as well. The formation of the abstraction created the advantage of third generation Programming Languages over the first and second generation of Programming Languages altogether.

However, this also saw the evolution of many first and second generations of Programming Languages to evolve as well as the example FORTRAN 77 is the evolved form that has seen its implementation in the second-generation Programming Language and uses the norms of the high-level coding criterion [34]. The earlier third generation Programming Language constituted of the COBOL, ALGO, and Fortran. Later after some duration C, C++, C#, JAVA basically was added in this generation. Some third-generation languages support structured programming and some of these support Object oriented programming. The advantage of that higher-level language is its user friendliness, feasibility in reading and writing.

2.4.4 Fourth Generation Programming Languages (4GL)

The era of 1970-90 saw the boom in the evolution of Programming Languages. The researchers had changed their focus from designing languages from the platform-based architecture, towards designing an overall Programming Language that could be used as the platform. The platform could be used to design and implement the commercial softwares. The fourth-generation Programming Language had been an evolved form in the implementation of abstraction as that the third-generation Programming Languages had been doing. The fact remains that the 4GL had been a step up in the form abstraction than

its predecessor 3GL. However, this does not change the overall matrix of C and JAVA languages going into the background as their importance was still there and the 4GL with their advance features held their own domains.

The era that persisted in the development of the fourth-generation Programming Languages were more of application development based. Which emphasized on the rapid development of software's as niche market of software development. Since developing applications in third generation Programming Language saw delays and were error prone therefore a new methodology was needed in this regard.

To create new application designers came up with idea of using higher level Programming Language and methodology, thus the fourth-generation Programming Language came to the horizon. The results showed that the generated codes were complex in nature as that created in the 3GL environment but were easier to implement and much easier to troubleshoot and more importantly were more reliable in creating. In general, the 3GL was closer to developing only whereas the 4GL and later generation that followed were more focused on speed of delivery and error resolutions. Examples of the fourth-generation Programming Languages are Ruby and Python.

2.4.5 Fifth Generation Programming Language (5GL)

The 4GL was followed by efforts to define and use a 5GL. The natural-language, block-structured mode of the third-generation Programming Languages improved the process of software development. The period of this generations starts from 1980's till onward. Moreover, after 1990 improvements in this generation were seen, systems were designed based on fuzzy logics, neural networks and in the field of artificial intelligence. The language involved parallel programming and artificial intelligence. That makes the computer thinking more like human being. Databases were programmed in a specialized manner to reduce the efforts of programmers and increase the involvement of system for that build in and enhanced libraries were introduced which minimize programmer effort to generate new applications. All the updated versions of high-level languages like C, C++, JAVA, .Net were also included in this version as well. Modern languages, especially the newly designed high-level Programming Languages are constructed to resolve the domain-

specific issues and requirements [35].

As a matter of fact, the fifth generation of Programming Languages are not Programming Languages at all but are in general algorithms designed to be problem specific and in a wider scope more generic in nature which learn from the data they are being exposed to and then design a methodology to resolve the issues. This has taken the program out of the hand of the programmer and the computer algorithms is now in command of resolving the issues. Today we are looking an intelligent system that are designed to minimize the human effort in problem solving and allowing the computer to be more real time-oriented system, some of the applications that the 5GL have found its roots into are the internet of things IoT, machine learning, data mining and artificial intelligence.

Some of the algorithms that are being used for this purpose are deep learning algorithms, which include Neural networks along with other algorithms such as the Naïve Bayes algorithms, decision trees, principal component analysis PSO, Markov chains.

2.4.6 Next Generations of Programming Languages

With the above-mentioned generations being discussed, the avenue of evolution of Programming Languages is still open with self-learning algorithms. These self-learning algorithms getting enhancements. The robotics geeing new technological edges, the options of evolution in Programming Language is bright.

2.5 Programming Languages

Number of Programming Languages were introduced in different eras. However, we are only focusing some of the commonly known languages.

2.5.1 FORTRAN

FORTRAN can be considered as the survivor of Programming Languages, when introduced for the first time the FORTRAN was considered as the Low-Code Platform. With the constant evolution and the changes made to it, FORTRAN has moved from 1GL to more advanced generations. Now we see that FORTRAN is in the forefront of Programming Languages. Fortran was introduced in the early 50’s and is still being used since many new derivatives of this Programming Language have come into exitance. The

developers tend to use them, as seen for the assembly language that shall be discussed in the later parts. The Fortran was envisioned as the machine level language. However, the later years have shown many iterations with many new version follow-ups have made this Programming Language still a market share holder [21] [36]. Since the syntax and their semantics have played a vital role in the development of the Programming Language, as seen in the development of Fortran, thus before we move on to the higher generations and discuss them let us first go through these points as well.

2.5.2 Assembler / Compiler

Assembly language is the basis of the second generation of Programming Language. The second generation were based on assembler and compilers to make the code much easier to write and understand by the programmer. Therefore, many of the assembly languages found their main stand in the practical fields of the embedded system design in which a microcontroller was designed using the second-generation Programming Languages. What is important to note in this regard is the introduction of transistors over the vacuum tubes this increased the programming and computation speeds of the system [37]. The overall impact was that the assembler had an influence in the market and the related Programming Languages were the game changers in this regard. Some of the codes that were also referred as the instructions that assembly languages used and were translated by the assembler are discussed in the Table 4, which is as follows.

Table 4. The assembly language instructions and their translated machine level code [34]

Sr #	Instruction in Assembly Language	Instruction meaning	Assembler converted Instruction
1.	ADD	Add values	32_{10}
2.	ADDU	Add unassigned values	33_{10}
3.	SUB	Subtract values	34_{10}
4	AND	Logical AND operation	36_{10}
5.	OR	Logical OR operation	37_{10}

6.	XOR	Logical exclusive OR operations	38_{10}
7.	NOR	Logical negated OR operation	39_{10}
8.	SLT	Set on less than	42_{10}
9.	MUL	Multiply	24_{10}
10.	DIV	Divide	26_{10}

In terms of embedded systems programming, the first language which echoes is the assembly language. The assembly language is still one of the sought-after Programming Language in the market. The importance of assembly language and its potential has caused different languages such as C language and JAVA to comply with the assembly language-based compilers as well.

2.5.3 Assembly Language

Introduced with second generations of Programming Languages the assembly language has made in routes in the programming fertility. The assembly language is a Low-Code Programming Language and with the advanced versions showing flavor of high-level Programming Language. The assembly language is used in a conditional manner in an environment which is mostly dependent of the platform. In case, if 8086 microcontrollers can be programmed by any assembly language, which shows the platform dependency of the assembly languages [38][39]. When designing a code on the assembly code it is a common practice to follow the flowchart in which the after designing the logical framework the second phase is to write a pseudocode which is same as the algorithm which shows the overall flow of data and how it is used to design the actual code.

```
; Example 2: Assembly Language
;
EXAMPLE2: MOV   DPTR,#50H       ;init pointer to 0050H
          MOV   R7,#0           ;init count = 0
REPEAT:   MOVX  A,@DPTR         ;char = @pointer
          INC   DPTR            ;increment pointer
IF:       CJNE  A,#'0',$+3      ;if char >= '0' AND
          JC    UNTIL           ;
          CJNE  A,#'9'+1,$+3    ; char <= '9'
          JNC   UNTIL           ;
THEN:     INC   R7              ;then increment counter
UNTIL:    CJNE  A,#0,REPEAT     ;char is 00H
          MOV   A,R7            ;store count in acc
HERE:     SJMP  HERE
          END                   ;example 2
```

Fig. 3. Example of Assembly Language [34]

Therefore, to cover it, the assembly language is used for the purpose of making the ease in the low-level programming becoming much easier and reliable in designing as the troubleshooting becomes easier with the introduction of the assembler [39]. The above shown example describes the classical example of the low-level code. In the low-level code the instructions are being used such as move and repeat. These instructions are human understandable and they are allowed to be used with registers such as the A , B which are referred as the registers. The registers hold the data used for computations and then their predefined instructions like END and BEGIN. They have their own predesigned instructions hardcoded in the assembler, along with all of this the assembler transforms. This complete code and narrates it into machine specific code also called as the gray code or the opcode.

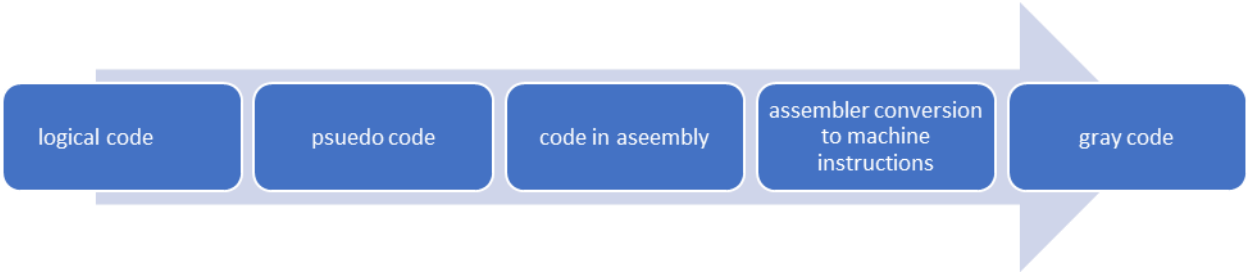

Fig. 4. Flowchart showing the steps undertaken to design the assembly code from start to finish [34]

2.5.4 C Language

The C language is one of the most used Programming Language. C language is used for

the development of application for industries ranging from defense to economic, from petrochemical to control systems. C language has shown it splat form independence and is also one of the most sought-after Programming Language which is used as assembly language program to be a platform independent high-level Programming Language as well. The C language is in general terms a more general-purpose Programming Language due to its vast portfolio. C language is used in personal computers, super computers and now in IoT. [40][41][42].

The inventor of C language was Dennis Ritchie at Bell Labs and its first major role was to implement the UNIX operating system. The facts that the C language was platform independent which is the main feature of the third-generation Programming Language also the backward retractability of C also employs it to be used as the assembly language program used for programming microcontrollers and supercomputers as well. In many ways, the concept of language going obsolete cannot be predicted in the far and near future.

2.5.5 JAVA Language

In terms of third generation languages, JAVA is one of those languages that were presented in this generation. Moreover, JAVA language is also popular in latest generations due to it's object-oriented approach. By the term object-oriented approach, it refers to the concept of how the code can be used to define a certain object in a class and how that object can be rephrased and defined to work for the nested objects to provide a functionality that can be optimized and is redundant to any kind of errors. Also, as being the third-generation Programming Language, the JAVA is also one of the Programming Languages which has the feature of abstraction in its compilers as predefined product [43][44]. Along with that, the main concept of the platform independent platform is also a critical feature of JAVA and thus it is also one of the used Programming Language as of C and C++. It is also worth mentioning that the syntax of JAVA is much same as that of C and C++[45].

Along with that, the discussion of functionality of JAVA is also very interesting. The fact that JAVA uses the compilers that are referred as JAVA Virtual Machine. The JAVA code

is compiled as bytecode and this code is then run on any machine without any need of compiler. Therefore, once a bytecode is created the need of using a compiler is not there. Most of the games based in the 90's was made in JAVA as the developers found its free ware architecture and compilers as easy going and imaginative base for designing new dimensions in gaming.

Fig 2.2. Atari console courtesy [Atari Japan co] was the platform that used JAVA as the source code for its games [46]

As of today, JAVA is being used as the web-based client-server-based Programming Language which is being used by millions of developers worldwide since, it is one of the most developed Programming Language in sense of controlling the developed codes [47].

JAVA was originally developed by James Gosling under the Sun microsystems-based labs which was later dissolved in to Oracle corporation. Although the syntax of JAVA is derived mainly form c language but unlike the C language the JAVA did not had any proper functional control for the low-level programming.

2.5.6 PYTHON

One of the most advanced Programming Language of the fourth-generation Programming Language is Python whose latest release was scheduled in 2, May 2018 according to the official website [55]. Python is based in interpreted high-level language which finds its use as general-purpose Programming Language [45][47]. Python is one of those language that is being used as the next generation of Programming Language as well [48]. With the machine learning trends coming into play and most of the application today being

developed are more business oriented and are intelligent therefore Python has find its application s in this field as well.

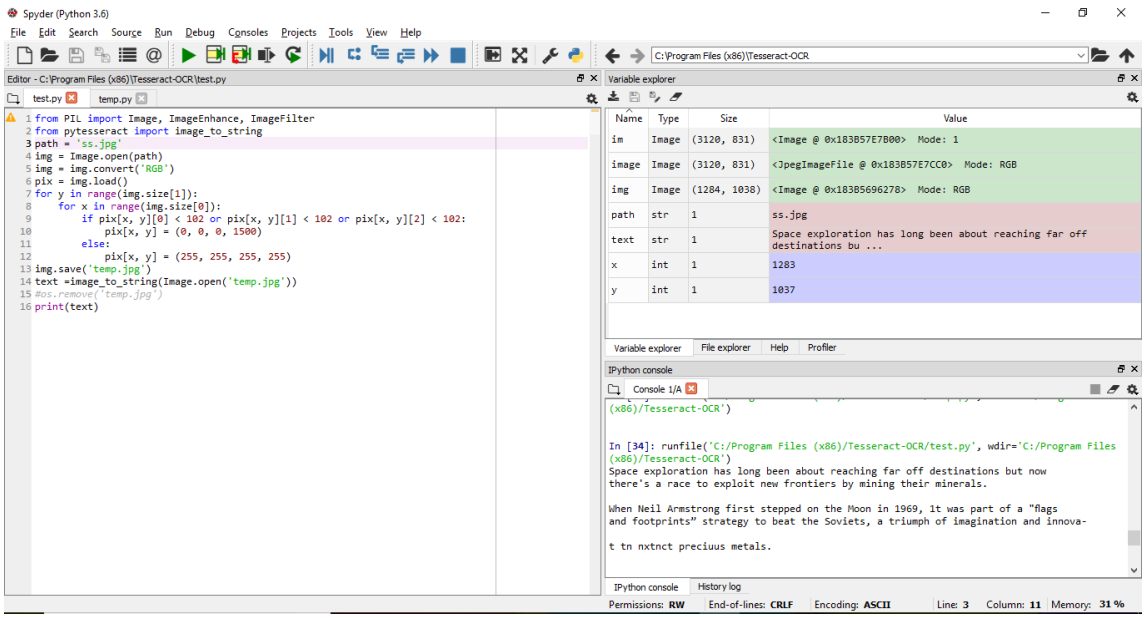

Fig. 5. Python code that can read and detect human hand-written text and then convert it into a word format [49]

The features that are introduced in Python are as follows [50]

- Open source
- Dynamic type system
- Standardized library
- Automatic memory management system
- Supports multiple programming paradigms such as
    - Object oriented
    - Imperative
    - Functional
    - Procedural

2.6 RAPID APPLICATION DEVELOPMENT AND LOW-CODE PLATFORM

As the applications demand was increasing dramatically, so to meet that demand Rapid Application Development (RAD) was introduced. RAD is used with the need of achieving

the development in minimum time possible [51][52][53]. Latterly, Low-Code Platforms where introduced to meet the massive demand of applications. The concept of Low-Code Platform is to develop the applications with minimal human effort and maximum speed.

2.6.1 Rapid Application Development

The RAD development usually consists of a team of 5 to 8 members who are mostly developers and business managers who have the sound knowledge of the development and have the authority to make design decisions [53][54][55][56]. There are number of RAD implementation methods and they all constitute in the latest trends in application developments. Windows XP is one of the examples of RAD development as well.

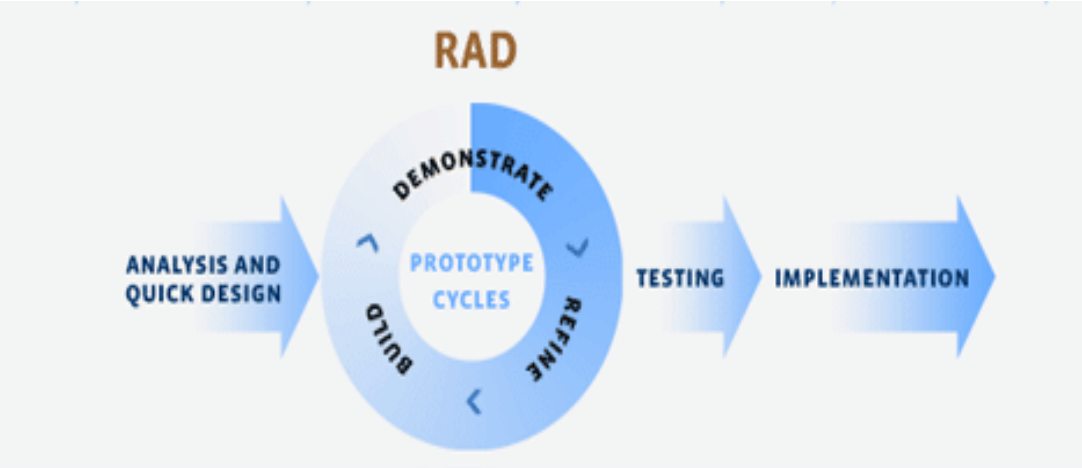

Fig. 6. Rapid Application development workflow [57]

Rapid application development has many advantages one such is the flexibility of programming and changing the programmability as the coding process continues. Also referred as the "Try it before you buy it" approach, is one of the latest approaches in the development of Programming Languages. The main advantage of RAD is to analyze and build the prototype model test it and change it within no time. The advantage that RAD brings is the speed [9] [58].

With many advantages of RAD there are many issues that need to be seen as well. The critics of RAD programming point out its lack of object oriented approach which has made its usability scarce [8][59]. The RAD had been the most emphasized design approach as it was the foundation on which the Low-Code Programming approach had been created and

designed as well.

2.6.2 Low-Code Programming

Low-Code development / programming platforms are mostly used in sense of developing the applications with minimal human effort required which caters of the view that applications that are developed in a rapid method. Low-Code Development Platforms (LCDP) are based on graphical user interface in designing the application as opposed to the hard-coded programming techniques. The feature of LCDP focuses in the development of the following [9],

- Databases
- Business processes
- User interface (web-based applications)

Low-Code Programming technique is basically derived from fourth generation programming ideology along with the concepts of RAD kept in sight. The Low-Code Programming enables the programmer to think less on the syntax of the code and put more emphasis in designing the esthetics and functionality of the application allowing lesser time required on troubleshooting and implementing. allowing to develop entirely operational applications [10].

Low-Code is generally referred to as novel and much new concept in the field of programming with the idea being pitched in 2011 alone. The Low-Code Programming is the next big thing and most of applications today developed through Low-Code Programming Languages.

The approaches that have created the Low-Code Programming are as follows

- Model driven software development approach
- Rapid application development
- Automatic code generation
- Visual programming

In a way, the above-mentioned techniques have all been merged together to achieve the Low-Code Programming technology that is in front of us. Some of the Low-Code Programming Languages are as follows.

- Salesforce
- Microsoft PowerApps
- Mendix
- Google App Maker
- TrackVia
- Appian

Globally the Low-Code Programming is getting famous and lot of new developers have found their interest in this programming technique. The reason for this evolution in programming is that the tedious task of writing down the syntax and then troubleshooting it was more laborious and time consuming.

2.7 Assessment criteria for Programming Language / Platform

Software or Platform evaluation criteria are not clearly defined and elaborated in the literature. The criteria meanings are open to the authors and they make their own interpretation. The exact meaning of a criterion is open to the evaluator's own interpretation. Sometimes, the terminology introduced for criteria by one author is different from the other author [62].

Before selecting any Programming Language or Platform, the main and the most important thing to understand is software life cycle. It depends on specifications, design, implementation and the maintenance. To select any Programming Language or Platform, these points should be considered as compulsory.

There are four important assessments by using them a Language can be considered as trusted if it has readability, write ability, reliability and better cost. Readability is the level of ease how easily a language can be understood and written in a well-mannered format. Writability is that how easily a language can be written efficiently in the given platform. A program of a language is considered as a reliable if it performs all the specific requirements under all the conditions. The cost is more important fundamental while choosing a language that how much expenses will come to train a programmer for that

language and fulfill all the requirements. At that point calls attention to that numerous components should be well-thought-out while choosing a Programming Language or platform, formal criteria can be considered even before. While a portion of variables will be incorporated into the given rule, these rules may likewise be utilized according to a pre-assessment device to limit the decisions field. One of the researchers demonstrates that although there are exceptionally solid sentiments regarding the matter of first Programming Languages[60] . Moreover, there may have any hard-observational tests schemes looking at changed Programming Languages or platforms. Another researcher [60] concurs regarding discussions finished with the relative benefits of various Programming Languages were very normal, there are moderately couple of test correlations of the ease of use of various Programming Language. The examination led, a one-to-one correlation for the effect for the achievement of two languages understudies in the very start of programming class is played out. Although the requirement of the given approach is an educator capable in the two languages and is not effectively stretched out to more than two languages at any given moment [60][61].

Nonetheless, it calls attention to that while episodic proof from early on programming courses is broadly accessible and singular language highlights have been contemplated from an intellectual perspective, the assurance of which language ought to be utilized for training starting programming remains a combative issue. One clarification offered for the shortage of studies is that the inconstancy among software engineers would render any examination useless. Another is that the contrasts between spaces are excessively awesome, making it impossible to play out a significant report that cannot help contradicting the commence that such investigations are impractical however does not offer any arrangements. Looking at languages is a troublesome errand, particularly when the languages do not have a similar worldview. Building up legitimate criteria for the examination is troublesome not just because an issue of what to quantify exists, yet in addition claiming the criteria may support one language over another attention is called to that in instructive settings the requests of different courses and educational module make it problematical, if certainly feasible, to look at changed languages. Further, extraordinary courses for the most part have adequately unique targets to make language correlations basically futile [61].

The factors which are to be considered as are noted before, numerous components must have been considered preceding starting any language process determination. Although these components in the end might be incorporated into the assessment criteria, they may likewise be utilized to limit the field of decisions. For instance, the determination will more likely than not properly guided by approach or worldview becoming educated and the quantity of languages should be utilized all through the arrangement of programming courses. Another choice is whether the office wishes to utilize a genuine Programming Language or an altered educating language. At last, changing cost Programming Languages should be considered [63].

The Methodology or the worldview the choice of Programming Language or platform worldview, which figures out what ought should have instructed, must go before the choice of the principal of Programming Language or the platform, which impacts how to show it. The worldview characterizes the structure inside which the understudies are educated. Programming ideal models are separated by the ideas that they underscore. Basic programming accentuates strategies working on (exemplified) factors; protest arranged programming underscores techniques working on (embodied) objects; practical programming underlines capacities working on changeless qualities; rationale programming stresses predicates working on permanent qualities; simultaneous programming accentuates forms trading messages fights that the focal thought while assessing initial courses of programming, ought not to be the languages yet rather the choice of a hypothesis or strategy of programming to instruct Attention on language is misinformed in light of the fact that it powers an accentuation on the mechanics of communicating key thoughts as opposed to concentrating on the key thoughts themselves. Extra contemplations incorporate whether language develops ought to be gained independently from or simultaneously with program configuration, how to decide the suitable harmony in between the programming within the vast and short-term programming, and whether a solitary language ought to be instructed all through the programming course grouping rather than numerous languages. The related issue of a solitary language versus numerous languages is another basic choice. The quantity of languages to cover in a course or educational modules includes an exchange off amongst broadness and profundity. A few instructors like to accommodate less languages in more

depth, learning enough about the languages in accepting understudies to have the capacity to utilize. Criteria for the determination of a Programming Language or the platform for nontrivial programs, while different teachers endeavor to indicate understudies the broadness of the programming field by presenting the understudy to however many languages as would be prudent. [64].

Selection to a language to perform all the specific requirements is too technical task. A decision must be made between proficient review and modified languages. An expert review language is one utilized as a part of industry and is instructed completely (except for extremely propelled highlights), by means of an economically accessible condition. An expert review language like Java or C++ furnishes understudies with involvement with a genuine situation. Any issues experienced are those that will be experienced, all things considered, getting ready understudies better for their expert lives. Tweaked languages, for example, Eiffel researcher and skill researcher isolate taking in programming from taking in the points of interest of a specific language, limiting the specialized issues that may divert from the learning procedure. When understudies take in the essentials, they can apply their programming learning to any language. [65]

Changing cost of a few instructors innocently imagine there is little change in cost in the Programming Languages because not having genuine or reliable outcomes if a choice turns on. In any case, that extensive overhead happens in receiving a specific language, including readiness of address materials, creating tasks and understudy works out, assessing and learning improvement situations and introducing the picked language, asking for and assessing course readings, and preparing faculty. Such overheads show that care ought to be taken while picking a language since that decision is probably going to affect the instructor for quite a long while.

2.7.1 Assessment Criteria

In the assessment criteria, 10 criteria were selected, and those criteria were StartUp experience, User Interface Development (UI), Data management, Digital Process Automation, Reporting and Analysis, Application Design & Development (AD&D) Support and Governance tools, Cloud Platform attributes, Vision, Road map, and Market Approach [72]. To evaluate vendors based on these criteria's, business developers and

power user were targeted.

Table 5. Criteria for Low-Code Platform assessment [72].

Criteria	Platform Evaluation Details
Startup Experience	Is there support from the platform for business developers to develop simple, single and straightforward application? Is the helping material or training easily available and accessible?
User Interface development	Is there any tool to create UX without coding? Is the UX including devices and screen resolution and layouts possible without coding?
Data Management	Are there data modeling tools available in the platform? Is the platform providing CRUD operations support?
Digital Process Automation	Is there declarative workflow development experience in the product? Does the product also address events and notifications in the workflow experience?
Reporting and Analysis	Is there support for dynamics reports in the platform? Does the platform provide dashboards and OOTB reports?
AD&D support and governance tools	Is there any tool provided by vendor to provide support and governance?
Cloud Platform Attributes.	Is there self-service public cloud provided by vendor? Is there any public cloud security certificate which covers PCI Level 1, SOC2 Type II' annually updated, FedRAMP, EU Privacy Shield, and HIPPA
Vision	Does the vendor vision meet the needs of the client, to become customer, serve and remain customer?

Road Map	How much stable, vendor has execution road map? How vendor achieve milestone, frameworks and bench makers which are defined in their strategy? Does the vendor have enough skills and resources to meet up their road map?
Market Approach	Is there any success-story go-to-market approach? Can vendor provide evidence of Revenue growth, vertical market strategy, Account momentum, and commercial model?

The Table 5 shows the ten criteria, where each criteria has its own evaluation details. The evaluation details have multiple questions to form the assessment of the Low-Code Platform.

3 WHY PLATFORM SELECTION IS IMPORTANT?

Decision making is a highly critical issue as it is directly associated with the origination's future. In the past we can be seen the success and failure stories while selection of right decision / technology or wrong decision / technology.

3.1 Success / Failure Stories

Since the genesis of man and since man has stepped on earth he has either succeeded of failed in accomplishing his goals. He succeeded in clothing himself for his protection against climate, discovered fire to warm him, build houses to shelter him, and with the advancement in science during recent years have enabled civilized man to developed sophisticated equipment without which life seems to be impossible, it is all because he has learnt from his failures. A very wise man and inventor Thomas Edison said, "I start where the last man left off". This shows the importance of not giving up after failure.

3.1.1 Success Stories

3.1.1.1 The success of Cell Phone

In the April of 1973, inventor, Mr. Martin Cooper, took the mobile phone technology to new level he took his machine to New York, held a demonstration to news reporters and the audience where a phone call was made while standing on Sixth Avenue [66]. Telecom has become a billion-dollar industry and its success is quite evident.

3.1.1.2 Success of the Xerox

The xerographic procedure, which was changed into invention by Chester Carlson in year 1938 was evolved and commercialized by using the Xerox business enterprise, is extensively used to supply fantastic textual content and image photos on paper. Carlson at the start referred to as the system electrophotography. It is primarily inspired from herbal phenomena: that substance of contrary electric quantization attracts MR Chester also concluded that some things are better conductors of current when comes in contact of light. Carlson came up with six-step technique to transfer pictures and images between surfaces to any other the use of these phenomena.

3.1.1.3 Success of touch screen

In 1971, a "touch sensor" became advanced by way of health practitioner Sam Hurst (founding father of electrographic) whilst he was a teacher at the college of Kentucky. This sensor known as the electrograph turned into patented via the university of Kentucky studies foundation.

The "electrographic" become no longer obvious like contemporary contact screens, however, it became a huge milestone in contact display era. The electrographic became decided as an industrial research as one of the 100 most vast new technical merchandise of the 12 months 1973 [67].

In 1974, the first proper contact display incorporating a transparent floor got here at the scene developed with the aid of Sam Hurst and Elographics. In 1977, electrographic advanced and patented a resistive touch screen era, the most popular contact screen technology in use nowadays.

In 1977, Siemens business enterprise financed an attempt by means of electrographic to produce the first curved glass contact sensor interface, which became the first device to have the call "touch screen" connected to it. On February 24, 1994, the organization formally modified its call from electrographic to Elo touch systems.
Touch screen is a product of programmable sensors. The sensors are made in such a way that they detect slightest of weight and make movement according to the weight.

3.1.2 Failure Stories

3.1.2.1 Failure of hydrogen Air ship

On the eve of 6th May 1937, the sky over New Jersey saw a tremendous fire, people shouted for help from a height of more than 100 feet's? It was fire everywhere. The era of the hydrogen air ship had come to an end. The accident was a great failure in aviation history. Man power could not do any good to avoid such a catastrophic event [68]. Great human loss was recorded 97 lost their life including both the passengers of the flight as well as crew members. The world watched in complete shock as millions of dollars and human lives lay ruined in ashes. From that very moment till this day, this technology has

been completely neglected by the masses as it turned out to be a major failure causing a very high death toll.

3.1.2.2 Failure of jet pack

Throughout the 1950s American defense forces wanted a jet pack for their solders to move easily around the war zone. A very talented scientist MR Wendell F. Moore of the famous Bell Aero System technology decided to peruse this dream of theirs [69]. After years of research and working, they concluded that they could not sustain a flight for more than 20 seconds and the cost of both developing a full-fledged jet system and fuel was very high. The other major disadvantage of this system was exposure of the solder to the enemy. All the reasons pointed towards the fact that this project must be stopped. A whopping 20 million dollars were utilized in the process. American defense forces also closed the project because of its badly failing. Pentagon opened its files and enabled Hollywood to use the system for its James Bond movies.

3.1.3 Lesson Learnt from Success / Failure Stories

Technology and its purposeful products are visible all around us. They form an integral part of our life. The impact of technology on society is both positive and negative with making life easier it has also helped in the creation of modern weaponry and defense systems. Not all are that lucky some technologies lead to a complete disaster and result in the loss of both money and precious lives. A technology such as hydrogen Air ships in the early part of twentieth century was a complete failure resulting in great amount of man loss. Many other technologies were also great failures, which resulted in great economic loss. So, taking the right decision is very important while selecting the technology.

3.2 Strategy Decisions

There are mainly two types of decisions, strategic decision and executive decision. Three main questions come under the umbrella of strategic decision and exactly three questions come under executive decision [70].

3.2.1 Strategy Decisions

1. How much should we spend on IT?

This is very first and critical question. To answer this question the IT goal needs to be crystal clear. A slight deviation from the goal can leads to major damage to the organization. There are always doubts that either executives are putting too much on IT or they are spending below the bar on IT.

The IT goals differ from organization to organization. In The successful organization, the senior management figure out the IT strategic role in the organization and according to that they decide how much spend on IT to achieve the objectives.

2. Which business processes should receive our IT dollars?

In this question, organization needs to pin point the business departments which needs IT services. Instead of putting the burden of all organization on IT department. There is always a need to avoid over burdening the IT department by assigning the irrelative projects.

The senior management avoid taking decision to prioritize the projects and leaves the decision in the hands of IT managers to decide the prioritizes. This thing will over burden the IT department as every department will claim that their project is more important. This will create the backlog at IT department end which eventually leads to the disaster. Sometimes, organizations prefer to hire third party IT companies as they do not trust their own IT department. However, they are responsible themselves as their lack of involvement.

3. Which IT capabilities should be firmwide?

Now, most of the executives have recognized the strategic and cost saving benefits that are because of centralizing the IT capabilities and introducing the standardize IT structure throughout the organization. Because of this thing, the executives are given the leverage to the IT managers to take decisions about what will be standardized and centralized, and what will not be standardized and centralized. Normally, IT managers take any of two approaches, depending on the culture of the organization. One approach is that they

standardized everything as it will save cost. The second approach is that the importance to the autonomous of the departments and give them exceptions in the standardizing. The second approach is expensive as compare to the first one [70].

3.2.2 Execution Decisions

1. How good do our IT services need to be?

It is totally depending on the executives, how much IT services they want to incorporate in their system. Different organizations required different characteristics, some demand reliability where as some demand accuracy. Some organizations demand top of the line services, for example, banks can not compromise on their data. They can not afford to lose date if the stock system crashed. They need hundred percent recovery [71].

2. What security and privacy risks will we accept?

Security and privacy is one of the main services of IT systems. However, spending too much on security will end up with great inconvenience. So moderate approach needs to be adopted for security and privacy. However, it is the senior executes duty to decide up to what level they need security.

3. Whom do we blame if an IT initiative fails?

Generally, it is observed that people think that the failure is just because of IT department. However, there are multiple factors involved. It is the business executives how changes the business structure.

It is therefore required that senior managers need to assign business executives to take responsibility for realizing the business benefits of an IT initiative. During the IT infrastructure development, they should keep in contact with the IT department. They also need to arrange trainings and orientations for the users. They should monitor that either they are getting business values from the new IT system. Only blaming the IT department will demoralized them and limit their capabilities. On the other hand, IT department needs continuous commitment from the managers and users who will use and get benefits from the system.

As for the responsibility is concerned, IT department should take responsibility to deliver the system on time and well within the budget. The delivered system should be capable enough to meet the organization's business requirements. However, the business executives should be held responsible for making changes in the organization to generate the business revenue by using the new system.

4 ASSESSING VENDORS AND LOW-CODE PLATFORM SELECTION

Low-Code Platforms are popular in Business Developers and Application Design & Development (AD&D). Initially, there was a perception that Low-Code Platforms are used only by the business developers. However, it is the surprising element that AD&D developers are also using Low-Code Platforms to develop and deliver the applications [72]. Low-Code Platforms for Business Developers are often used by IT Developers [72].

4.1 Assessing Vendors

Keeping in value the popularity of Low-Code Platforms in business developers as well as in AD&D, the assessment is carried out in this research for Low-Code Platforms for business developers and AD&D separately based on the available literature.

For the assessment process, 13 Low-Code Platform providers were selected [72]. These are the providers, Microsoft, Quick Base, Caspio, Zudy, FlowForma, KissFLOW, Filemaker, Zudy, Matssoft, Vizru, Intellect, Kintone, and Nintex.

The Table 5 shows the ten criteria, where each criteria has its own evaluation details. The evaluation details have multiple questions to form the assessment of the Low-Code Platform. Based on the assessment, which is made in section 2.7, the Table 6 is formed which shows overview of the 13 Low-Code vendors.

Table 6. Quick overviw of vendor assessment [72]

Company	Startup Experience	User interface development	Data Management	Digital Process Automation	Reporting and Analysis	Support and Governance	Cloud Platform Attributes	Market Approach	Vision	Road Map

Quick Base	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
MatsSoft	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
Caspio	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
Microsoft	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
Nintex	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
FlowForma	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
FileMaker	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
KiSSFLOW	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
Zudy	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
Pulpstream	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
Kintone	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
Vizru	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
Intellect	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●
● ● ● Differentiated				● ● ● On Par				● ● ● Needs Improvement		

Table 6 shows the quick overview of 13 vendors' standings for 10 assessment criteria. The possible grades are, from highest to lowest, "Differentiated", "On Par" and "Needs Improvement". "Differentiated" is the highest grade and "Needs Improvement" is the lowest grade. The standings of the vendors can be seen in three standards "Differentiated", "On Par" and "Needs Improvement".

4.1.1 Quick Base

Quick Base Platform is very good in startup experience, data management. Quick Base's market approach and vision is highly supportive. However, Quick Base needs to improve the User Interface (UI) as still UI of Quick Base is not responsive. Moreover, Quick Base needs to improve in Digital Process Automation (DPA) [72]. In the assessment, Forrester named Quick Base as "Leader" [72].

Over all Quick Base performance is highly satisfactory as it allows the users to create

application within a day. The assessment of all the features can be seen in the table below.

Table 7. Quick Base Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 7 illustrates that the Startup Experience, Support and Governance, Cloud Platform Attributes, Data management, Market Approach, Vision, Reporting and Analysis, and Road Map criteria for Quick Base are in "Differentiated" position. However, the User Interface development, and Digital Process Automation criteria for Quick Base are in "On Par" and "Needs Improvement" positions respectively.

4.1.2 MatsSoft

MatsSoft Low-Code Platform is good in UX, Reporting and Analysis. However, in the areas of startup experience and data management, MatsSoft performance is average. In the assessment, Forrester named MatsSoft as "Leader". The assessment of all the ten features can be found in the table below.

Table 8. MatsSoft Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance

● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 8 depicts that the User interface development, Reporting and Analysis, Support and Governance, Cloud Platform Attributes, Vision, and Road Map criteria for MatsSoft Low-Code Platform are in "Differentiated" position. Conversely, Data management, Startup Experience, Digital Process Automation, and Market Approach criteria are in "On Par" position.

4.1.3 Caspio

Caspio Low-Code Platform provides strong UI, data management and Reporting features. However, Caspio does not have process designer. Lacks in support, and vision [72]. In the assessment, Forrester named Caspio as "Leader" [72]. The assessment of Caspio features can be found in the table below.

Table 9. Caspio Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision

● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 9 demostrates that the User interface development, Data management, Reporting and Analysis, Cloud Platform Attributes, Road Map, and Market Approach criteria for Caspio Low-Code Platform are in “Differentiated” position. Moreover, Startup Experience, and Vision criteria are in ”On Par” position. However, Digital Process Automation is on ”Need Improvement” position.

4.1.4 Microsoft

Microsoft’s Low-Code Platform, PowerApps has strong features of UX, cloud, and DPA. However, PowerApps Low-Code Platform is less effective reporting, startup, and support [72]. In the assessment, Forrester named Microsoft as “Strong Performer”. Assessment of PowerApps’s all features can be seen in the table below.

Table 10. Microsoft Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 10 shows that the User interface development, Digital Process Automation, Cloud Platform Attributes, Market Approach, and Vision criteria for Microsoft Low-Code Platform are on “Differentiated” position. Moreover, Data management, and Support and Governance criteria are on ”On Par” position. On the other hand, Reporting and Analysis criteria is on ”Need Improvement” position.

4.1.5 Nintex

Nintex Low-Code Platform has strong features of startup experience, DPA, and support. However, it needs to improve UI and cloud features. Moreover, Nintex does not have strong data management feature to its users [72]. In the assessment, Forrester named Nintex as “Strong Performer” [72]. The assessment of Nintex’s features can be seen in the table below.

Table 11. Nintex Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 11 demostrates that Startup Experience, Digital Process Automation, Support and Governance, and Market Approach criteria for Nintex Low-Code Platform is in “Differentiated” position. Moreover, User interface development, Cloud Platform Attributes, Reporting and Analysis, and Road Map criteria are on ”On Par” position. On the other hand, Data management criteria is on ”Need Improvement” position.

4.1.6 FlowForma

FlowForma Low-Code Platform has strong startup experience, DPA, and UX features. However, data management, and cloud features are not very strong, these features are just average features of FlowForma. Moreover, FlowForma needs more improvement in reporting, and support features [72]. In the assessment, Forrester named FlowForma as "Strong Performer" [72]. The assessment of all the feature of FlowForma can be found in the table below.

Table 12. FlowForma Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 12 illustrates that the Startup Experience, User interface development, Digital Process Automation, and Road Map criteria for FlowForma Low-Code Platform are in "Differentiated" position. Moreover, Data management, Cloud Platform Attributes, Market Approach, and Vision criteria are in "On Par" position. However, Reporting and Analysis is on "Need Improvement" position.

4.1.7 FileMake

Data management, reporting, and support are the strong features of FileMaker Low-Code Platform. FileMaker just has average startup experience. Moreover, FileMaker needs to

improve its UX, and DPA features [72]. In the assessment, Forrester named FileMaker as "Contender". The assessment of FileMaker features can be found in the table below.

Table 13. FileMaker Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 13 shows that the Data management, Reporting and Analysis, and Support and Governance criteria for FileMaker Low-Code Platform are in "Differentiated" position. Moreover, Startup Experience, Market Approach, and Vision criteria are in "On Par" position. However, User interface development, Digital Process Automation, and Cloud Platform Attributes are on "Need Improvement" position.

4.1.8 KiSSFLOW

KiSSFLOW Low-Code Platform has excellent feature of startup experience. The assessment shows that UI, and cloud features of KiSSFLOW are average. Data management and reporting features of KiSSFLOW needs improvements [72]. In the assessment, Forrester named KiSSFLOW as "Contender" [72]. The assessment of all the features of KiSSFLOW can be found in the table below.

Table 14. KiSSFLOW Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 14 depits that the Startup Experience criteria for KiSSFLOW Low-Code Platform is in “Differentiated” position. Moreover, User interface development, Digital Process Automation, Support and Governance, Cloud Platform Attributes, and Market Approach criteria are in ”On Par” position. However, Data management , Reporting and Analysis, Vision, and Road Map are on ”Need Improvement” position.

4.1.9 Zudy

Data management and Vision features of Zudy Low-Code Platform are excellent. However, DPA, and reporting features of Zudy are just average. Moreover, UI, cloud, and road map features need improvement [72]. In the assessment, Forrester named Zudy as “Challenger”. The assessment of Zudy’s features can be found in the table below.

Table 15. Zudy Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach

● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 15 illustrates that the Data management, and Vision criteria for Zudy Low-Code Platform are in “Differentiated” position. Moreover, Digital Process Automation, Reporting and Analysis, and Support and Governance are in ”On Par” position. However, Startup Experience, User interface development, Cloud Platform Attributes, Market Approach, and Road Map Data management are on ”Need Improvement” position.

4.1.10 Pulpstream

Pulpstream Low-Code Platform’s UX, data management, and cloud features are average features. Moreover, startup experience and reporting features need improvement. In the assessment, Forrester positioned Pulpstream Low-Code Platform as “Challenger” [72]. In the assessment, Forrester named Pulpstream as “Challenger” [72]. The assessment features of Pulpstream can be found in the table below.

Table 16. Pulpstream Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map

● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement

The Table 16 shows that no criteria for Pulpstream Low-Code Platform is in “Differentiated” position. However, User interface development, Data management, Digital Process Automation, Cloud Platform Attributes, Market Approach, and Vision are in ”On Par” position. Moreover, Startup Experience, Reporting and Analysis, Support and Governance, and Road Map are on ”Need Improvement” position.

4.1.11 Kintone

Kintone Low-Code Platform startup experience and data management features are average. Moreover, UI and vision features need improvement. Forrester positioned Kintone as “Challenger” in the assessment [72]. The assessment features of Kintone can be found in the table below.

Table 17. Kintone Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 17 demonstrates that no criteria for Kintone Low-Code Platform is in “Differentiated” position. However, Startup Experience, Data management, User interface

development, Data management, Digital Process Automation, Digital Process Automation, Reporting and Analysis, Support and Governance, Cloud Platform Attributes, Road Map are in ”On Par” position. Moreover, User interface development, Market Approach, and Vision are on ”Need Improvement” position.

4.1.12 Vizru

Vizru Low-Code Platform’s DPA feature is strong for the user. However, UI, cloud and startup experience features need improvement. In the assessment, Forrester named “Challenger”. The assessment features of Vizru can be found in the table below.

Table 18. Vizru Low-Code Platform vendor assessment [72]

● ● ● Startup Experience	● ● ● Support and Governance
● ● ● User interface development	● ● ● Cloud Platform Attributes
● ● ● Data management	● ● ● Market Approach
● ● ● Digital Process Automation	● ● ● Vision
● ● ● Reporting and Analysis	● ● ● Road Map
● ● ● Differentiated ● ● ● On Par ● ● ● Needs Improvement	

The Table 18 shows that Digital Process Automation criteria for Vizru Low-Code Platform is in “Differentiated” position. However, Reporting and Analysis, and Road Map are in ”On Par” position. Moreover, Startup Experience, Data management, Support and Governance, Cloud Platform Attributes, Market Approach, and Vision are on ”Need Improvement” position.

4.1.13 Intellect

Intellect Low-Code Platform has good UI, UX features. However, most of its features need more improvement such as data management, cloud, and DPA. In the assessment, Forrester named Intellect as “Challenger” [72]. The assessment features of Intellect can be found in the table below.

Table 19. Intellect Low-Code Platform vendor assessment [72]

● ○ ○ Startup Experience	● ○ ○ Support and Governance
● ● ○ User interface development	● ○ ○ Cloud Platform Attributes
● ○ ○ Data management	● ○ ○ Market Approach
● ○ ○ Digital Process Automation	● ○ ○ Vision
● ○ ○ Reporting and Analysis	● ○ ○ Road Map
● ● ● Differentiated ● ● ○ On Par ● ○ ○ Needs Improvement	

The Table 19 illustrates that no criteria for Intellect Low-Code Platform is in “Differentiated” position. However, User interface development is in ”On Par” position. Moreover, Startup Experience, Data management, Digital Process Automation, Reporting and Analysis, Support and Governance, Cloud Platform Attributes, Market Approach, Vision and Road Map are on ”Need Improvement” position.

4.2 Decision Tree for Selection of Low-Code Platform

In the Low-Code Platforms industry, there are number of Low-Code Platforms are available, however, all platforms do not have equal features. Some platforms have flexible architecture; however, some have full featured packages but have inflexible architecture. Some companies need Low-Code Platforms that can be customized to fulfill their business

needs. Some companies need Low-Code Platforms that have fully features packages that perfectly matches the needs of the company's businesses. Some companies have Technical employees and some companies have Non-Technical employees. So, they need right Low-Code Platform for their employees, who will use the platform to meet their business needs.

Different companies have different type of employees as well as they are working on different type of applications. The decision tree will help the companies to find out the most suitable Low-Code Platform for their company. This decision tree is constructed based on the data found in the literature review [4][72][73][74][75][76][77].

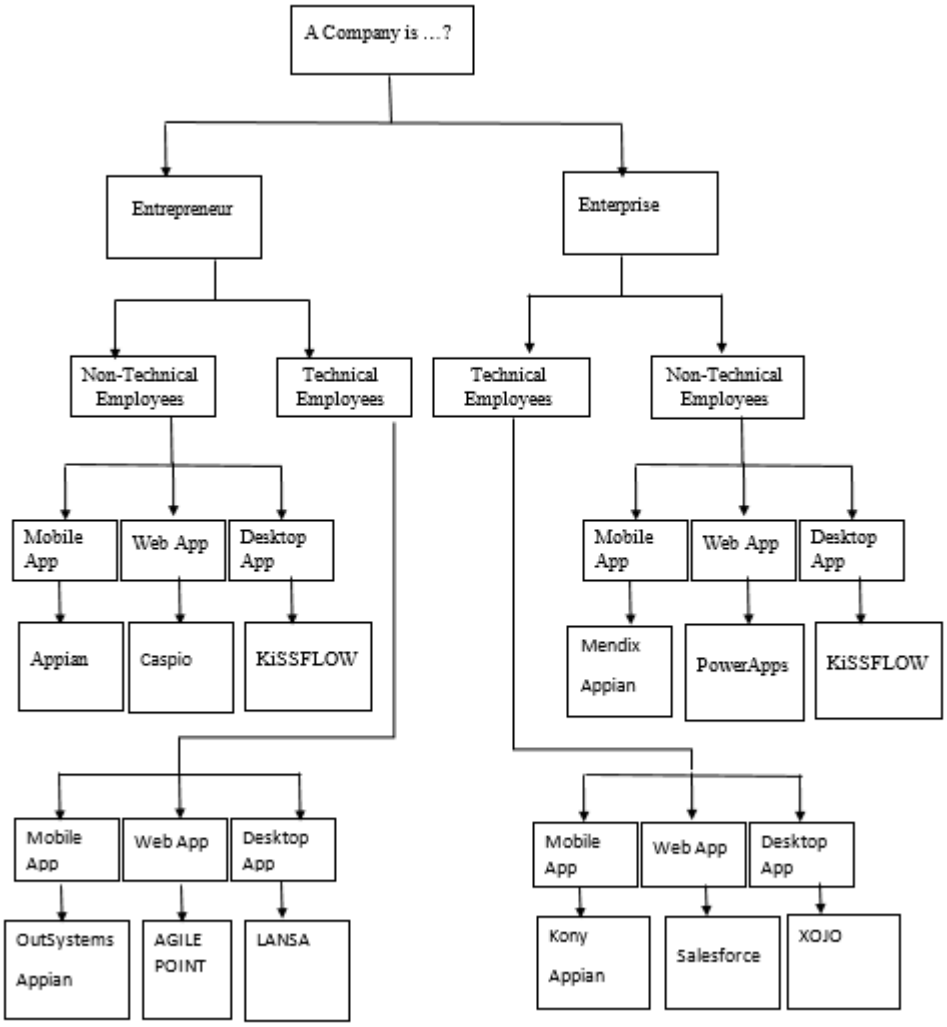

Fig. 7. Decision tree for selection of Low-Code Platforms

In the decision tree, the researcher discussed about two main terminologies, the first one is the organization and the second one is the application. To understand the structure of the decision tree, the definitions of these terminologies need to be defined.

4.2.1 Organization

“An organized group of people with a particular purpose, such as a business or government department” [78].

In the thesis, the researcher discussed the following two types of the organizations

4.2.1.1 Entrepreneur

“An entrepreneur is an individual who founds and runs a small business and assumes all the risk and reward of the venture” [79].

“A person who sets up a business or businesses, taking on financial risks in the hope of profit” [80].

“An entrepreneur is an individual who, rather than working as an employee, founds and runs a small business, assuming all the risks and rewards of the venture. The entrepreneur is commonly seen as an innovator, a source of new ideas, goods, services and business/or procedures” [81].

4.2.1.2 Enterprise

“A company organized for commercial purposes; business firm” [81].

“The activity of providing goods and services involving financial and commercial and industrial aspects” [82].

4.2.2 Application

“An application is any program, or group of programs, that is designed for the end user.

Applications software (also called end-user programs) include such things as database programs, word processors, Web browsers and spreadsheets" [83].

"Application software is a type of computer program that performs a specific personal, educational, and business function. Each program is designed to assist the user with a particular process, which may be related to productivity, creativity, and/or communication" [84].

"Application software is a program or group of programs designed for end users" [85].

In the thesis, the researcher focused three types of applications i.e. Mobile App, Web App, and Desktop App.

4.2.2.1 Mobile App

"A mobile app is a software application developed specifically for use on small, wireless computing devices, such as smartphones and tablets, rather than desktop or laptop computers" [86].

"A mobile application, most commonly referred to as an app, is a type of application software designed to run on a mobile device, such as a smartphone or tablet computer" [87].

4.2.2.2 Web App

"A web application is a computer program that utilizes web browsers and web technology to perform tasks over the Internet" [88].

"A Web application (Web app) is an application program that is stored on a remote server and delivered over the Internet through a browser interface" [89].

4.2.2.3 Desktop App

"An application that runs stand-alone in adesktop or laptop computer" [90].

The term "desktop software" refers to software that Runs on a Mac, Windows or Linux desktop computer that is set up for use in one place such as a desk [91].

4.3 Cases

In this section, the discussion is about the decision tree and how the 12 cases are formulated based on the application and company types. This is the most critical part of the thesis where the researcher is discussing about how these 12 cases are made. These cases are based on the organization types and application types. For these cases two organization types are selected i.e. Entrepreneur and Enterprise and three applications types i.e. Mobile App, Web App, and Desktop App. Now, the question is that why the researcher made exactly 12 cases. The answer is fairly simple as the researcher selected most common organization types as well as most common application types. So, with the combination of organization types and application types these 12 cases are formulated which will be discusses one by one.

4.3.1 Case 1

In the Case 1, a company is an Entrepreneur and has Non-Technical Employees who requires Mobile App. For the case 1, Appain Low-Code Platform is most suitable.

4.3.1.1 Appian

Appian is one of the leading Low-Code Platforms for digital transformations. Appian is equally suitable for entrepreneur as well as enterprises. Appian has features of easy to change, secure and unified processes [77]. The world’s leading innovative companies are using Appian to implement their business applications. Different sectors of the industry using Appian Low-Code Platform such as Automotive and Manufacturing, Energy, Financial services, Banking and Capital Markets, Healthcare Players and Providers, Insurance, Life Sciences and Pharma, Public Sector, Retail, Telecom and Media, Transportation.

Table 20. Price structure of Appian Low-Code Platform [92].

Trial Edition	Application Edition		Enterprise Edition
Free (Trial)	90 $ (Per user/ Per month)		180 $ (Per user/ Per month)
Test drive Appian's leading Low-Code app development platform	Business users quickly create and collaborate on a single application		Harness the power and speed of Appian across all your enterprise applications
Separate and secure cloud instance	All platform functionality is available		Unleash digital transformation throughout your organization without artificial limits
All platform functionality is available			All platform functionality is available

The Table 20 shows the price structure of Appian Low-Code Platform. The price structure contains three types of prices i.e. Trail Edition which is free, Application Edition which is US$ 90 per user per month, and Enterprise Edition which is US$ 180 per user per month.

4.3.2 Case 2

In the Case 2, a company is an Entrepreneur and has Technical Employees who requires Mobile App. In this case, Outsystems and Appains both are suitable. Appian is already discussed in the above section.

4.3.2.1 OutSystems

OutSystems provides the Platform as a Service (Paas). OutSystems provides the visual development interface to develop entire application. Outsystems provides the facility to integrate newly built application to the existing application. OutSystems created

application can also be customized by using our own code [93].

As OutSystem is a Paas products that's runs in cloud, so the hardware requirements for the machines are low, however a high-speed connection of internet is required.

Outsystems is the number one Low-Code platform [94]. Outsystesms has thousands of worldwide customers. It is equally good for entrepreneurs as well as enterprises. Outsystems can be a good choice for entrepreneurs as it allows to build free app for up to 100 users.

The big giants of industry are using OutSystems. A few of them are Honda Motor Co., AXZ, ING Group, Intel Corporation, Mercedes-Benz, Siemens AG, Vodafone, Volkswagen and many more. Not only industry using OutSystems, educational institutions are also using OutSystems such as Kent State University, Bob Jones University, and University of Georgia. As for educational institutions, Bob Jones University, Georgia Institute of Technology, Kent State University and University of Georgia are already using this software.

Table 21. Price structure of OutSystems Low-Code Platform [95].

Free	Enterprise	Universal
Free Forever	Starting at US$5,400/month (Billed Annually)	Starting at US$12,250/month (Billed Annually)
Application Capacity: Limited	Application Capacity: Scaled to needs with additional capacity available	Application Capacity: Scaled to needs with additional capacity available
End User Capacity: Up-to 100	End-User Capacity: Starts at 100	End-User Capacity: Unlimited
Dev Environments: 1	Dev Environments: 3+ Environments (Dev / Test / Prod)	Dev Environment: 3+ Environments (Dev / Test / Prod)
OutSystems Cloud: Shared	OutSystems Cloud : Dedicated	OutSystems Cloud : Dedicated

Community Support: Yes	On-premises and Private / Public / Hybrid Cloud deployment options	On-premises and Private / Public / Hybrid Cloud Deployment Options
	8x5 Support (24x7 available)	8x5 Support (24x7 available)
	Standard Customer Success Program included	Standard Customer Success Program included

The Table 21 illustrates the price structure of OutSystems Low-Code Platform. The price structure contains three types of prices i.e. Free which is free forever, Enterprise which is starting at US$ 5,400/month, and Universal which is Starting at US$ 12,250/month.

4.3.3 Case 3

In the Case 3, a company is an Entrepreneur and has Non-Technical Employees who requires Web App. For Case 3, Caspio is the most suitable Low-Code Platform.

4.3.3.1 Caspio

Caspio is a Low-Code Platform for non-technical developers to develop web applications. It's is completely non-coding Low-Code Platform with the basic fee 59$ per month. So, it is eventually the most suitable choice for entrepreneurs who have non-technical developers. However, caspio offers corporate level pricing packages, so that can be the choice of enterprises who have non-technical developers.

Table 22. Price structure of Caspio Low-Code Platform [96].

Basic	Professional	Performance	Corporate
59$ per month (Free Trial)	249$ per month (Free Trial)	249$ per month (Free Trial)	1,699$ per month (Free Trial)
Starter plan with live customer support	Features and services for most scenarios	More apps, more capabilities, more productivity	More apps, more capabilities, more productivity

Users: Unlimited	Users: Unlimited	Users: Unlimited	Users: Unlimited
DataPages: 5	DataPages: 50	DataPages: 150	DataPages: 400
Data Transfer: 500 MB	Data Transfer: Unlimited	Data Transfer: Unlimited	Data Transfer: Unlimited
Live Support: Limited	Live Support: Business Support	Live Support: Business Support	Live Support: Premium Support and SLA
Features: Core Features	Features: Standard Features	Additional Features: Yes	Premium Features: Yes Expert Session
	Expert Sessions: 2 Hours	Expert Sessions: 5 Hours	Expert Sessions: 10 Hours
			Sub-Accounts: Yes

The Table 22 depicts the price structure of Caspio Low-Code Platform. The price structure contains four types of prices i.e. Basic, Professional, Performance, and Corporate. All these types have different prices. However, they have free trail all the types.

4.3.4 Case 4

In the Case 4, a company is an Entrepreneur and has Technical Employees who requires Web App. For this case, Agile Point Low-Code Platform is most suitable.

4.3.4.1 AGILE POINT

AgilePoint is a Low-Code platform, which exists with the slogan of "Low-Code. No limits". AgilePoint is an innovative Low-Code platform for making automation and digital apps in a quick and easy way.

Forrester Research placed AgilePoint among top five Low-Code platform in all Low-Code segments. Moreover, Forrester Research placed AgilePoint in one of the top five Low-Code platform, which provides the integration of the apps with Microsoft tools [76].

In the Business Process Management (BPM), AgilePoint NX is the leading Low-Code platform, which is used in cloud deployment in more than 25 countries.

AgilePoint provides its services to different sectors such as energy, finance, manufacturing, insurance, government, and healthcare. The main clients of the AgilePoint are CKW, Ex Arte, GRENKE, Katoen Natie, Nyasys, RICOH and TRES60. All these are well known companies, which shows the strength of AgilePoint Low-Code platform.

For enterprise applications, AgilePoint NX makes it easy by connection system, content and the people. This ease is handy, as AgilePoint can run on any known tools such as Salesforce, Dropbox, One Drive, Google Drive, Office 365, and SharePoint. Moreover, AgilePoint Apps can be easily integrated with any legacy enterprise systems. In this way, the customer can build their applications in half of the time as compare to the time they will take to make their application in the traditional way.

Table 23. Price structure of AgilePoint Low-Code Platform [97].

	Named Seat (minimum 50 seats)	**Concurrent Seat (minimum 20 seats)**	**Unlimited Seat (minimum 2 Cores - production)**
	Per Seat(Monthly)	Per Seat(Monthly)	Per CPU Core(Monthly)
AgilePoint NX Editions	Sharepoint | Enterprise	Sharepoint | Enterprise	Sharepoint | Enterprise
Apps 1-10	10$ | 25$	65$ | 85$	2,950$ | 3,950$
Apps 11-100	12$ | 29$	75$ | 125$	3,950$ | 5,950$
Apps 101-Unlimited	18 | 35$	95 | 165$	5,950$ | 7,950$

The Table 23 illustrates the price structure of AgilePoint Low-Code Platform. The price structure contains three types of prices i.e. Named Seat, Concurrent Seat, and Unlimited Seat. All these types have different prices with minimum number of seats.

4.3.5 Case 5

In the Case 4, a company is an Entrepreneur and has Non-Technical Employees who requires Desktop App. For this case, KiSSFLOW Low-Code Platform is the most suitable.

4.3.5.1 KiSSFLOW

KiSSFLOW is suitable for non-technical employees to develop desktop applications as it' is easy to use platform. To develop desktop applications in KiSSFLOW, no coding is required. KissFLOW is equally good for entrepreneurs as well enterprises.

KiSSFLOW has more than 10,000 clients. MOTOROLA, The Telegraph, Domino's, MAERKS, Pepsi, AIRBUS, British Council, The University of Chicago, CASIO, MECHELIN and HubSpot are potential customers of KiSSFLOW.

Table 24. Price structure of KiSSFLOW Low-Code Platform [98].

Standard Edition	Bulk Pricing	Non-Profit Pricing
Per Seat(Monthly)	Per Seat(Monthly)	Per CPU Core(Monthly)
9 $ user/month	Users > 100 Price: Custom	Special pricing for education and not-for-profit organizations

The Table 24 shows the price structure of KiSSFLOW Low-Code Platform. The price structure contains three types of prices i.e. Standard Edition, Bulk Pricing, and Non-Profit Pricing. All these types have different prices.

4.3.6 Case 6

In Case 6, a company is an Entrepreneur and has Technical Employees who requires Desktop App. For this case, LANSA Low-Code Platform is the most suitable.

4.3.6.1 LANSA

LANSA Low-Code platform is suitable for both entrepreneur and enterprises for Desktop applications. Moreover, LANSA also provides mobile and web apps development. LANSA offers its one tool for Windows, Web and Mobile application development with

only one price.

LANSA provides services to different sectors, Allianz, ADVANCE, AHC, Actinver, ABCS Loan Systems, Alpura, Beacon Insurance, Bidfood, Blackmaker, CGA, DAIHATSU, Eagle Systems, HONDA, Kawasaki, Porsche, VISA Card, Kellogg's, MoMA, and High Liner are the main clients.

Table 25. Price structure of LANSA Low-Code Platform [99].

EC2 Instance type	Software / Year	EC2 / hr
t2.micro	$2,813	$0.016
t2.small	$2,813	$0.032
t2.medium	$2,813	$0.064
t2.large	$2,813	$0.121
m4.large	$2,813	0.192
m4.xlarge	$2,813	0.384
m4.2xlarge	$2,813	0.768
m4.4xlarge	$2,813	1.536
m4.10xlarge	$2,813	3.84
m4.16xlarge	$2,813	6.144

The Table 25 show the price structure of LANSA Low-Code Platform. The price structure contains two types i.e. Software per year, and Elastic Compute Cloud (EC2) per hour.

4.3.7 Case 7

In Case 7, a company is an Enterprise and has Non-Technical Employees who requires Mobile App. In this case, both Mendix and Appians are suitable. Appian is already

discussed in the above section.

4.3.7.1 Mendix

Mendix is one of the most popular Low-Code development platforms. Mendix provides support in three type of applications i-e Event Driven Applications, Data Driven Applications and Process Oriented Applications. Over 3,400 worldwide organizations using Mendix. Mendix provides the 10 times faster development as compare to the traditional development [100].

Mendix Low-Code Platform provides the visual modeling tools that helps non-technical developers to develop mobile apps in a fast way. Mendix powered by IBM to bring Low-Code Platform development on the cloud.

ING, KLM, MERCK, Action for children, Arch, bam, Dsi, Eurail, GAPLESS, iss Facility Services, Saga, ProRail, Springer Healthcare, TNT, and CHUBB are few of the Mendix's customers.

Table 26. Price structure of Mendix Low-Code Platform [101].

	Community	Single App	Pro	Enterprise
	Free	Starting at €1.875/month	Starting at €5.375/month	Starting at €7.825/month
No. of Applications		1	Unlimited	Unlimited
Public Mendix Cloud	✓	✓	✓	✓
On-premises				✓
Private Cloud				✓
Uptime Guarantee		99.50%	99.50%	99.50%
Support	Community	Gold	Gold	Platinum
Automated Backup		✓	✓	✓
Horizontal Scaling				Sub-Accounts: Yes
Fail Over				✓

Support for Continuous Integration & Development				✓

The Table 26 shows the price structure of Mendix Low-Code Platform. The price structure contains four types of prices i.e. Community, Single App, Pro, and Enterprise. Moreover, Community price is free.

4.3.8 Case 8

A company is an Enterprise and has Non-Technical Employees who requires Mobile App. In this case, Kony and Appians are suitable. Appians is already discussed in the above section.

4.3.8.1 Kony

Kony is a “Leader” In Mobile Low-Code Development Platforms [102]. Moreover, Kony was named as commanding position in the market with the products as are accepted as best of breed [103]. Kony has the slogan, “Transforming your. Today and Tomorrow”.

Kony is the high demanding among the customers in all over the world. Kony, has more than 350 protentional successful customers in 45 countries. The major ABN-AMRO, aenta, ALFA insurance, ally, CIBC bank, Nationwide, TOYOTA, Shred-it, KPMG, and SGN.

Unfortunately, the pricing structure of Kony is not available.

4.3.9 Case 9

In Case 9, a company is an Enterprise and has Non-Technical Employees who requires Web App. For this case, PowerApps Low-Code Platform is the most suitable.

4.3.9.1 PowerApps

PowerApps is a Low-Code platform provided by Microsoft, which enables the citizen and IT developers to create business apps for browser as well as for phone or table. In PowerApps, no coding experience is required [104]. Microsoft`s PowerApps is a

combination of PowerPoint and Excel, where PowerPoint provides drag and drop features and Excel provides logical expressions.

Table 27. Price structure of PowerApps Low-Code Platform [105].

PowerApps for Office 365	PowerApps Plan 1	PowerApps Plan 2	PowerApps for Dynamics 365
Included in select Office 365 Plans	$7 per user per month	$40 per user per month	Included in select Apps and Dynamics 365 Plans
Extend the capabilities of Office 365 (SharePoint Online, Teams, Excel and more)	Everything included with PowerApps for Office 365	Everything included with PowerApps Plan 1	Everything included with PowerApps Plan 2
Create and run canvas apps learn more	Create and run canvas apps that connect to a wide range of data sources using premium connectors	Create and run model driven apps learn more	Extend Dynamics 365 or create custom model-driven and canvas apps with the full functionality of PowerApps Plan 2
Create automated workflows with Microsoft Flow	Create and run canvas apps built on the Common Data Service for Apps	Create and manage instances of Common Data Service for Apps	Access restricted Dynamics 365 entities and APIs learn

The Table 27 depicts the price structure of PowerApps Low-Code Platform. The price structure contains four types of prices i.e. PowerApps for Office 365, PowerApps Plan 1, PowerApps Plan 2, and PowerApps for Dynamics 365. PowerApps. All these types have different prices.

4.3.10 Case 10

In Case 10, a company is an Enterprise and has Technical Employees who requires Web App. For this case, Salesforce Low-Code Platform is the most suitable.

4.3.10.1 Salesforce

Salesforce App Cloud is designed to extend company's Customer Relationship Management (CRM). Salesforce is offering a full-fledged app and component marketplace and a veritable arsenal of visual app development environments and tools for average business users and developers alike. Salesforce has the largest app marketplace and most extensive set of tools.

Table 28. Price structure of Salesforce Low-Code Platform [106].

Employee Apps Starter	Employee Apps Plus	Heroku Enterprise Starter
€ 25 user/month (billed annually)	€ 100 user/month (billed annually)	Starting at € 4,000 company/month (billed annually)
✓ Custom app development with access to 10 objects per user	✓ Custom app development with access to 110 objects per user	✓ Support for modern open-source languages
✓ Point-and-click app development with Lightning	✓ Point-and-click app development with Lightning	✓ Smart containers and elastic runtime
✓ Community for employees	✓ Community for employees	✓ Simple horizontal and vertical scalability

✓ Account and contact management	✓ Account and contact management	✓ Services add-on ecosystem
✓ Task and event tracking	✓ Task and event tracking	✓ Fine-grained access controls
✓ Employee cases	✓ Employee cases	✓ Shared application portfolio
✓ Workflow and approvals	✓ Workflow and approvals	✓ Enterprise account team
✓ Knowledge base	✓ Knowledge base	✓ Salesforce Identity federation
✓ Native collaboration (Chatter)	✓ Native collaboration (Chatter)	✓ Team and user administration
✓ Salesforce Identity	✓ Salesforce Identity	✓ Resource utilization management
✓ Customizable reports and dashboards	✓ Customizable reports and dashboards	✓ Trusted application operations
✓ Mobile development kit	✓ Mobile development kit	✓ Salesforce data synchronization
✓ Integration via real-time APIs	✓ Integration via real-time APIs	
✓ Cloud database	✓ Cloud database	
✓ Assets and work orders	✓ Assets and work orders	

The Table 28 shows the price structure of Salesforce Low-Code Platform. The price structure contains three types of prices i.e. Employee Apps Starter, Employee Apps Plus, and Heroku Enterprise Starter. All these types have different prices.

4.3.11 Case 11

In Case 11, a company is an Enterprise and has Non-Technical Employees who requires Desktop App. For this scenario, KissFLOW Low-Code Platform is most suitable.

4.3.11.1 KiSSFLOW

KiSSFLOW is an ideal choice for enterprises who have non-technical employees and they want to develop desktop applications. To develop desktop applications in KiSSFLOW, no coding is required. KissFLOW is equally good for entrepreneurs as well enterprises.
KiSSFLOW has more than 10,000 clients. MOTOROLA, The Telegraph, Domino's, MAERKS, Pepsi, AIRBUS, British Council, The University of Chicago, CASIO, MECHELIN and HubSpot are potential customers of KiSSFLOW.

The price structure of KiSSFLOW can been seen in Table 25.

4.3.12 Case 12

In Case 12, a company is an Enterprise and has Technical Employees who requires Desktop App. For this scenario, XOJO is the most suitable Low-Code Platform.

4.3.12.1 XOJO

XOJO is suitable for technical employees to develop desktop applications. Desktop applications can be developed for Windows, Mac and Linux. XOJO provides drag and drop facility to design user interface. XOJO provides its code editor to write your business logic. XOJO also provides Aps to create applications for cross platforms.

XOJO is not only for desktop applications for Windows, Mac and Linux. XOJO can be used to create mobile and web applications.

Google, AT&T, CISCO, CNN, DELL, hp, NASA, Adobe, Apple and intel are the users of XOJO [107].

Table 29. Price structure of XOJO Low-Code Platform.

Desktop	Pro	Enterprise
299 $ / Per Year	699 $/ Per Year	1,999 $ / Per year

All Desktop platforms: • Windows • Mac • Linux • Database Access	Everything in Desktop, plus: • Web • iOS • Console / Service Apps • Consulting Leads	Everything in Pro, plus: • Kickoff Meeting • Code Reviews • XOJO Advisory Board • Priority Support

The Table 29 illustrates the price structure of XOJO Low-Code Platform. The price structure contains three types of prices i.e. Desktop, Pro, and Enterprise. Desktop price is US$ 299/Year. Pro price is US$ 699/Year and Enterprise price is US$ 1,999/Year.

4.4 Low-Code Platforms and Their Vendors

In the decision tree, the researcher used different Low-Code Platform's Vendors than the vendors used in assessment criteria. These vendors are different because of limited literature availability to find out which Low-Code Platform is suitable for which type of organization and application type. The researcher created the decision tree on the best knowledge available in the literature till today. The selection is made on the market approach, experience, price, and vision towards the organization and application type.

Table 30. Low-Code Platforms and their Vendors

Platform	Vendor	Address
Appian	Appian	11955 Democracy Drive , Suite 1700 Reston, VA 20190, USA
OutSystems	OutSystems, Inc.	374 Congress St, 2nd Floor Boston, MA 02210, USA
Caspio	Caspio	2953 Bunker Hill Lane, Suite 201 Santa Clara, California, USA
Agilepoint	Agilepoint, Inc	1916B Old Middlefield Way Mountain View, CA 94043, USA

KiSSFLOW	Orange Space	800 West El Camino Real, Mountain View, CA 94040, USA
LANSA	LANSA Inc	2001 Butterfield Road, Suite 102 Downers Grove, IL, 60515, USA
Mendix	IBM	268 Summer Street Boston, MA 02210, USA
Kony	Kony	9225 Bee Cave Road, Building A, Suite 300, Austin, Texas 78733, USA
PowerApps	Microsoft	Microsoft Corporation One Microsoft Way Redmond, WA 98052-6399, USA
Salesforce	Salesforce	One Market Suite 300. San Francisco, CA 94105, USA
XOJO	XOJO, Inc	12600 Hill Country Blvd., Suite R-275 Austin, Texas 78738, USA

The Table 30 shows the Low-Code Platform and its vendor. Moreoever, addresses of vendors are also provided.

5 CONCLUSIONS

The decision tree in the chapter 4 is a comprehensive structure, which provides the facility to the organizations to select the most suitable Low-Code Platform based on their organization type as well as based on the application type. The decision tree is simple and compact to select the suitable Low-Code Platform. The decision tree is formed the 12 cases and these cases are formed based on the organization and application types. Now, it is easy for an organization to use this decision tree to select the most suitable Low-Code Platform. The assessment criteria in the chapter 4 provides the standings of the Low-Code Platform's vendor. In the assessment criteria, standings of 13 different Low-Code vendors can be seen which is calculated based on the 10 criteria.

In general, the literature about Low-Code Platforms specially the academic literature is rarely available. However, the available literature provided an idea about Low-Code Platform selection. The most authentic and comprehensive reports on Low-Code Platforms are of the Forrester reports, however these reports are not publicly free of cost available. This issue is due to the fact, that Low-Code Platform term is relatively new in the academics.

In the thesis, the researcher found the answers of the research questions, less research work is done in this area previously. Selection of Low-Code Paltform is not clearly defined and elaborated in the literature. However, based on the available literature, the researcher provided a decision tree to choose the suitable Low-Code Platform based on the Organization and application type. In addition, assessment criteria are provided to find out the standings of Low-Code Platforms.

We have seen the failure stories in the chapter 3. So, as a future direction, the same Decision Tree or the Decision Tree with changes can be used to select the other technologies to avoid the chances of failure.

The research is not only contributing in the industry to selection to suitable Low-Code Platform. It is also contributing in the academics by providing the numerous Low-Code Platforms and their vendors.

Furthermore, as future research, the research can be carry out in two ways. Firstly, more cases in the decision tree can be used by providing more organization types such Small Medium Enterprise (SME), and Large Enterprise. Secondly, more criteria can be added in the assessment criteria to provide more refined standings of the Low-Code Platform vendors.

REFERENCES

1. Information Age. (2018). *4 requirements of a Low-Code development platform.* [online] Available at: http://www.information-age.com/4-requirements-Low-Code-development-platform-123469520/ [Accessed 12 Mar. 2018].
2. Techopedia.com. (2018). *What is a Platform? - Definition from Techopedia.* [online] Available at: https://www.techopedia.com/definition/3411/platform [Accessed 12 Jun. 2018].
3. SearchServerVirtualization. (2018). *What is platform? - Definition from WhatIs.com.* [online] Available at: https://searchservervirtualization.techtarget.com/definition/platform [Accessed 12 Jun. 2018]
4. RICHARDSON, C. and Rymer, J.R., (2016). *The Forrester Wave™: Low-Code Development Platforms*. Available at: https://agilepoint.com/wp-content/uploads/Q2-2016-Forrester-Low-Code.pdf (Accessed: 10 May 2018).
5. dzone.com. (2018). *Why Developers Fear Low-Code - DZone DevOps.* [online] Available at: https://dzone.com/articles/why-developers-fear-Low-Code [Accessed 12 Jun. 2018].
6. UK, P., Looks, F., Projects, Z., Marvin, R. and Marvin, R. (2018). *Building an App With No Coding: Myth or Reality?*. [online] PCMag UK. Available at: http://uk.pcmag.com/zoho-projects/82699/feature/building-an-app-with-no-coding-myth-or-reality [Accessed 11 Apr. 2018].
7. Low-Code Basics. (2018). *Introduction to Low-Code Software | Low-Code Basics*. [online] Available at: https://www.appian.com/Low-Code-basics/introduction [Accessed 4 Mar. 2018].
8. Yu, D. (2011). Towards the Rapid Application Development Based on Predefined Frameworks. *Journal of Software,* 6(9).
9. Van Engelen, R., Whalley, D. and Yuan, X. (2004). Automatic validation of code-improving transformations on low-level program representations. *Science of Computer Programming,* 52(1-3), pp.257-280.

10. Vechev, M. and Yahav, E. (2016). Programming with “Big Code”. Foundations and Trends® in Programming Languages, 3(4), pp.231-284.
11. AgilePoint. (2018). *Gartner and Forrester define “Citizen Development” differently; here’s how- AgilePoint.* [online] Available at: http://agilepoint.com/gartner-forrester-define-citizen-development-differently-heres/ [Accessed 28 Apr. 2018].
12. Bergin, T. (2009). Jean Sammet: Programming Language Contributor and Historian, and ACM President. *IEEE Annals of the History of Computing*, 31(1), pp.76-85.
13. Lutz, M. (2002). System Architecture with XML [Book Review]. *Computer*, 35(10), pp.72-72
14. Broy, M. (2007). Editorial–Science of Computer Programming–25 years. *Science of Computer Programming*, 66(2), pp.103-104.
15. HOPL: History of Programming Languages Conference. (1979). *IEEE Annals of the History of Computing*, 1(1), pp.68-71.
16. Budd, T. (1991). Blending imperative and relational programming. *IEEE Software*, 8(1), pp.58-65.
17. Ciancarini, P. (1996). An overview of declarative process modelling using logic programming. *The Knowledge Engineering Review*, 11(04), p.303.
18. Haramis, G. (1992). Implementing a feasibility study “a procedural approach”. *Annual Review in Automatic Programming*, 16, pp.133-138
19. Wilson, Leslie B, and Robert G Clark., (2001) Comparative Programming Languages. *Harlow, England: Addison-Wesley*,. Print.
20. Donaldson, Alastair, and Vasco T. Vasconcelos. (2015). "Selected Papers On Programming Language Approaches To Concurrency And Communication-Centric Software (PLACES 2014)". *Journal Of Logical And Algebraic Methods In Programming* 84 (5): 683. doi:10.1016/j.jlamp.2015.06.004.
21. Wilkes, M. (1968). The outer and inner syntax of a programming language. *The Computer Journal*, 11(3), pp.260-263.
22. Languageexplained.com. (2018). *LanguageExplained.com*. [online] Available at: http://languageexplained.com/ [Accessed 9 Mar. 2018].

23. Shurmer, H.V. (1971). "Low-Level Programming For The On-Line Correction Of Microwave Measurements." *Radio and Electronic Engineer* 41.8: 357. Web.
24. Loidl, Hans-Wolfgang (2006). Trends In Functional Programming. *Bristol, UK: Intellect*. Print.
25. Young, S.J. (1980). "Low-Level-Device Programming With A High-Level Language." *IEE Proceedings E Computers and Digital Techniques* 127.2: 37. Web.
26. Horowitz, Ellis. (1984). Fundamentals Of Programming Languages. *Berlin, Heidelberg: Springer Berlin Heidelberg*. Print.
27. Melmed, Allan J. (2006). "Electrical Lead Into Vacuum Tubes." *Review of Scientific Instruments* 34.3 (1963): 307-308. Web.
28. Crawford, Tim et al. (2018) "Gigaom | The Industry Leader In Emerging Technology Research." *Gigaom.com*. N.p. Web. 15 June 2018.
29. Hall, C. and Clayton, R., (1979). Isolation of assembler coding 'optimizations'. *Software: Practice and Experience*, *9*(3), pp.248-248.
30. Dahl, V. and Saint-Dizier, P., (1985). Natural language understanding and logic programming.
31. McCurdy, C., (1987). The FORTH Programming Language for Control Systems: Potential Advantages. *Measurement and Control*, *20*(4), pp.45-48.
32. Clark, T., (2015). XPL: A language for modular homogeneous language embedding. *Science of Computer Programming*, *98*, pp.589-616.
33. Owlcation. (2018). *Types of Computer Languages with Their Advantages and Disadvantages.* [online] Available at: https://owlcation.com/stem/Types-of-Computer-Languages-with-Advantages-and-Disadvantages [Accessed 23 Jun. 2018].
34. Patton, B., (1993), June. Object-oriented Fortran 77 (a practitioner's view). In *ACM SIGPLAN Fortran Forum* (Vol. 12, No. 2, pp. 23-24).
35. High performance Fortran language specification. (1993). *ACM SIGPLAN Fortran Forum*, 12(4), pp.1-86.
36. An, T., Zhang, L. Y. Z. B. Z., & Lan, X. (2008). Assembly language programming.
37. Saint-Dizier, P. (1986). An approach to natural-language semantics in logic programming. *The Journal of Logic Programming*, 3(4), pp.329-356.

38. Holmes, N. (1997). A tale of assembly. Annals of the History of Computing, *IEEE*, 19(4), pp. 47-49.
39. Bergin, T. (2007). A history of the programming languages. *Communications of the ACM*, 50(5), pp. 69-74.
40. Super assembler. (1977). *Microprocessors*, 1(3), pp.197-200.
41. Ryan, R. (1985). C programming language and a C compiler. *IBM Systems Journal*, 24(1), pp. 37-48.
42. Black, A. (2004). Post-Javaism. *IEEE Internet Computing*, 8(1), pp. 93-95.
43. Danforth, S. and Tomlinson, C. (1988). Type theories and object-oriented programming. *ACM Computing Surveys*, 20(1), pp.29-72.
44. Khan, A. (2014). Comparative analysis of JAVA & C++ history, similarities & differences, syntax and design issues. *International Journal of Technology and Research*, 2(4), p. 131.
45. Sanner, M. F. (1999). Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1), 57-61
46. Daniels, J. (1996, July). Why RAD is Bad. In *Presentation at meeting of RESG. Imperial College, London*.
47. Grand, M., (1997). *Java language reference*. O'Reilly & Associates, Inc.
48. Kerr J and Hunter R (1994) Inside RAD: how to build fully functional computer systems in 90 days or less. *Mcgraw-Hill, New York*.
49. Recognition, P. (2018). *Python package for handwriting recognition*. [online] Code Review Stack Exchange. Available at: https://codereview.stackexchange.com/questions/68527/python-package-for-handwriting-recognition [Accessed 23 May 2018].
50. Loguidice, B. and Barton, M., (2014). *Vintage Game Consoles: an inside look at Apple, Atari, Commodore, Nintendo, and the greatest gaming platforms of all time*. Focal Press.
51. Madnani, N. (2007). *Getting started on natural language processing with Python. Crossroads*, 13(4), pp.5-5.
52. Martin, J. (1992) Rapid Application Development. *Prentice-Hall*, Englewood Cliffs
53. Reilly, J.P. and Carmel, E., (1995). Does RAD live up to the hype?. *IEEE Software*, 12(5), pp.24-26.

54. Boehm, B.W., (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), pp.61-72.
55. Rizwan, M. (2011). Application of 80/20 rule in software engineering Rapid Application Development (RAD) model. *Communications in Computer and Information Science*, 181(3), pp. 518-532.
56. Daud, N. M. N. (2010). Implementing rapid application development (RAD) methodology in developing practical training application system. Information Technology (ITSim), *International Symposium*, 3, pp. 1664-1667.
57. Ghahrai, A. (2018). *Rapid Application Development Model - RAD Model. Testing Excellence*. [online] Available at: https://www.testingexcellence.com/rapid-application-development-rad/ [Accessed 24 Apr. 2018].
58. Howard, A., (2002). Rapid Application Development: Rough and dirty or value-for-money engineering? *Communications of the ACM*, 45(10), pp. 27-29.
59. Lank, E., (2006). User centred rapid application development. *Rapid Integration Of Software Engineering Techniques*, 3943, pp. 34-49.
60. Fox, J., (2005). Rethinking second language admission requirements: Problems with language-residency criteria and the need for language assessment and support. *Language Assessment Quarterly: An International Journal*, *2*(2), pp.85-115.
61. Otto, K. and Hölttä-Otto, K., 2007. A multi-criteria assessment tool for screening preliminary product platform concepts. *Journal of Intelligent Manufacturing*, *18*(1), pp.59-75.
62. Jadhav, A. S., and Sonar, R. M. (2009). Evaluating and selecting software packages: A review. *Information and software technology*, *51*(3), pp.555-563.
63. O'Hagan, S., Pill, J. and Zhang, Y., (2016). Extending the scope of speaking assessment criteria in a specific-purpose language test: Operationalizing a health professional perspective. *Language Testing*, *33*(2), pp.195-216.
64. Parker, K.R., Ottaway, T.A. and Chao, J.T., (2006). Criteria for the selection of a programming language for introductory courses. *International Journal of Knowledge and Learning*, *2*(1-2), pp.119-139.
65. Goldberg, D.W. and Cockburn, M.G., (2010). Improving geocode accuracy with candidate selection criteria. *Transactions in GIS*, *14*, pp.149-176.

66. Kim Y, Hwang H., (2009). Incremental discount policy of cell-phone carrier with connection success rate constraint. *European Journal of Operational Research*.;196(2):682-687.
67. Interactive Touch Screen Display Technology | Avocor [Internet]. Avocor | different by design. 2018 [cited 1 July 2018]. Available from: https://www.avocor.com/
68. Hydrogen Airship Disasters | Airships.net [Internet]. Airships.net. 2018 [cited 1 July 2018]. Available from: http://www.airships.net/hydrogen-airship-accidents/
69. Jet pack fails Fox news broadcast [Internet]. CNET. 2018 [cited 1 July 2018]. Available from: https://www.cnet.com/news/jet-pack-fails-fox-news-broadcast/
70. Ross, J. W., & Weill, P. (2002). Six IT decisions your IT people shouldn't make. Harvard business review, 80(11), 84-95.
71. March, J.G. and March, J.G., (1988). *Decisions and organizations* (pp. 335-358). Oxford: Blackwell.
72. Rymer, J.R., (2017). *The Forrester New WaveTM: Low-Code Platforms For Business Developers*. [online] Available at: https://www.forrester.com/report/The+Forrester+New+Wave+LowCode+Platforms+For+Business+Developers+Q4+2017/-/E-RES140271 [Accessed 24 May 2018].
73. Stackify. (2018). *How to Choose the Best Low-Code Platform For Your Dev Team*. [online] Available at: https://stackify.com/Low-Code-dev-platform/ [Accessed 3 May. 2018].
74. Rymer, J.R., (2018). *Customers Illuminate The Benefits And Challenges Of Low-Code Development Platforms*. [online] Available at: https://www.forrester.com/report/Customers+Illuminate+The+Benefits+And+Challenges+Of+LowCode+Development+Platforms/-/E-RES141793 [Accessed 14 May 2018].
75. RICHARDSON, C, Rymer, J.R., (2014). *Five Customer-Facing Scenarios Shape "Low-Code" Platform Choices*. [online] Available at: https://www.forrester.com/report/Five+CustomerFacing+Scenarios+Shape+LowCode+Platform+Choices/-/E-RES117606 [Accessed 10 May 2018].
76. RICHARDSON, C, Rymer, J.R., (2016). *Vendor Landscape: The Fractured, Fertile Terrain Of Low-Code Application Platforms*. [online] Available at:

https://www.forrester.com/report/Vendor+Landscape+The+Fractured+Fertile+Terrain+Of+LowCode+Application+Platforms/-/E-RES122549 [Accessed 2 June 2018].

77. Sarah, M, (2017). *Measuring The Total Economic Impact™: The Benefits Of Appian's Low-Code Platform*. [online] Available at: https://www.appian.com/resources/measuring-total-economic-impact-benefits-appians-low-code-platform/ [Accessed 11 May 2018]
78. Oxford Dictionaries | English. (2018). *organization | Definition of organization in English by Oxford Dictionaries*. [online] Available at: https://en.oxforddictionaries.com/definition/organization [Accessed 16 Jul. 2018].
79. Staff, I. (2018). *Entrepreneur*. [online] Investopedia. Available at: https://www.investopedia.com/terms/e/entrepreneur.asp [Accessed 16 Jul. 2018]
80. Oxford Dictionaries | English. (2018). *entrepreneur | Definition of entrepreneur in English by Oxford Dictionaries*. [online] Available at: https://en.oxforddictionaries.com/definition/entrepreneur [Accessed 17 Jul. 2018].
81. Company, B. and Hopper, P. (2018). *Company, Firm, Enterprise,Business?*. [online] Englishforums.com. Available at: https://www.englishforums.com/English/CompanyFirmEnterpriseBusiness/gpvbn/post.htm [Accessed 17 Jul. 2018].
82. Vocabulary.com. (2018). *business enterprise - Dictionary Definition*. [online] Available at: https://www.vocabulary.com/dictionary/business%20enterprise [Accessed 17 Jul. 2018].
83. Webopedia.com. (2018). *What is an Application (Application Software)? Webopedia Definition*. [online] Available at: https://www.webopedia.com/TERM/A/application.html [Accessed 18 Jul. 2018]
84. Base, Q. (2018). *Application Software 101 | QuickBase*. [online] Quickbase.com. Available at: https://www.quickbase.com/articles/application-software-basics [Accessed 18 Jul. 2018]
85. Techopedia.com. (2018). *What is Application Software? - Definition from Techopedia*. [online] Available at: https://www.techopedia.com/definition/4224/application-software [Accessed 18 Jul. 2018]

86. WhatIs.com. (2018). *What is mobile app? - Definition from WhatIs.com*. [online] Available at: https://whatis.techtarget.com/definition/mobile-app [Accessed 18 Jul. 2018]
87. Techopedia.com. (2018). *What is a Mobile Application? - Definition from Techopedia*. [online] Available at: https://www.techopedia.com/definition/2953/mobile-application-mobile-app [Accessed 18 Jul. 2018]
88. Maxcdn.com. (2018). *What is a Web Application?*. [online] Available at: https://www.maxcdn.com/one/visual-glossary/web-application/ [Accessed 16 Jul. 2018]
89. SearchSoftwareQuality. (2018). *What is Web application (Web app)*? - Definition from WhatIs.com. [online] Available at: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app [Accessed 18 Jul. 2018]
90. Pcmag.com. (2018). *Desktop application Definition from PC Magazine Encyclopedia*. [online] Available at: https://www.pcmag.com/encyclopedia/term/41158/desktop-application [Accessed 18 Jul. 2018]
91. The Balance. (2018). *How Is Desktop Software Different From an App*. [online] Available at: https://www.thebalance.com/what-is-desktop-software-1293673 [Accessed 18 Jul. 2018]
92. Appian. (2018). *Appian Platform Pricing*. [online] Available at: https://www.appian.com/platform/pricing/ [Accessed 4 May 2018].
93. Outsystems.com. (2018). *The #1 Low-Code Platform for Digital Transformation | OutSystems*. [online] Available at: https://www.outsystems.com/ [Accessed 2 Apr. 2018].
94. Golovin, D. (2017). OutSystems as a Rapid Application Development Platform for Mobile and Web Applications.
95. Outsystems.com. (2018). *OutSystems Pricing & Editions*. [online] Available at: https://www.outsystems.com/pricing-and-editions/ [Accessed 4 May 2018].
96. Caspio. (2018). *Caspio Cloud Database Pricing | Review and Compare Plans*. [online] Available at: https://www.caspio.com/pricing/ [Accessed 4 May 2018].

97. AgilePoint. (2018). *Pricing - AgilePoint*. [online] Available at: http://agilepoint.com/pricing/ [Accessed 4 May 2018].
98. KiSSFLOW. (2018). *Pricing - KiSSFLOW*. [online] Available at: https://kissflow.com/pricing/ [Accessed 4 Jul. 2018].
99. Aws.amazon.com. (2018). *AWS Marketplace: Visual LANSA Development Environment Web application framework Win2012*. [online] Available at: https://aws.amazon.com/marketplace/pp/B0153VJXQW/ref=srh_res_product_title?ie=UTF8&sr=0-3&qid=1448445935571 [Accessed 19 Jun. 2018].
100. Mendix. (2018). *Application Platform As A Service - Application Development Platform*. [online] Available at: https://www.mendix.com/application-platform-as-a-service [Accessed 24 Mar. 2018].
101. Appian. (2018). *Appian Platform Pricing*. [online] Available at: https://www.appian.com/platform/pricing/ [Accessed 4 May 2018].
102. Jeffrey, S., (2017). *The Forrester Wave™: Mobile Low-Code Development Platforms*. [online] Available at: https://www.forrester.com/report/The+Forrester+Wave+Mobile+LowCode+Development+Platforms+Q1+2017/-/E-RES136055 [Accessed 10 Jun. 2018].
103. Kony. (2018). *Dedication, flexibility make Kony a Low-Code leader*. [online] Available at: https://www.kony.com/resources/blog/dedication-flexibility-make-kony-Low-Code-leader [Accessed 8 May 2018].
104. Docs.microsoft.com. (2018). *What are canvas apps? - PowerApps*. [online] Available at: https://docs.microsoft.com/en-us/powerapps/getting-started [Accessed 4 Jul. 2018].
105. Powerapps.microsoft.com. (2018). *Pricing - PowerApps*. [online] Available at: https://powerapps.microsoft.com/en-us/pricing/ [Accessed 4 Jul. 2018].
106. Salesforce.com. (2018). *Pricing*. [online] Available at: https://www.salesforce.com/eu/products/platform/pricing/ [Accessed 4 Jul. 2018].
107. Xojo.com. (2018). Xojo: *About Xojo*, Inc. [online] Available at: https://www.xojo.com/company/about.php [Accessed 11 Jun. 2018].
108. Techopedia.com. (2018). *What is a Developer? - Definition from Techopedia*. [online] Available at: https://www.techopedia.com/definition/17095/developer [Accessed 27 Jul. 2018].