RAPIDLY SCALING DIGITAL TRANSFORMATIONS OF HEALTHCARE SYSTEMS

LEVERAGING CLOUD-BASED LOW-CODE DEVELOPMENT PLATFORMS WITH DEVSECOPS GUIDELINES TO RAPIDLY SCALE THE DIGITAL TRANSFORMATION OF HEALTHCARE SYSTEMS

BY EKENE TITILOPE OLATUNJI, B.SC.

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree Master of Science in eHealth

McMaster University MASTER OF SCIENCE (2021) Hamilton Ontario (eHealth)

TITLE: Leveraging Cloud-Based Low-Code Development Platforms with DevSecOps Guidelines to Rapidly Scale the Digital Transformation of Healthcare Systems.

AUTHOR: Ekene Titilope Olatunji, B.SC

SUPERVISOR: Dr. Norman Archer
archer@mcmaster.ca

THESIS COMMITTEE MEMBERS:

Dr. Kamran Sartipi
sartipi@mcmaster.ca

Dr. Ridha Khedri
khedri@mcmaster.ca

Mr. Muhammad Khan
arfankhan77@yahoo.com

NUMBER OF PAGES: 80

Executive Summary/Abstract

The job of healthcare professionals in the healthcare sector has never been more critical than now due to the current unprecedented rate of long-term IT infrastructural changes and digital transformation. The 2019/2020 COVID-19 pandemic has been a major driver of these changes. Cultivating a culture of digital innovation and transformation is now at the forefront of the healthcare value-chain. There is an increased need to optimize the operations of the healthcare system, improve collaboration among Health Teams and deliver more agile and secure applications to support both clinical and administrative processes in healthcare institutions. These driving forces require a vision and strategy for digital transformation in the healthcare system, involving a closer look at modern DevSecOps best practices in the application development process. The fast-growing popularity of Cloud Computing has driven the consideration of Low-Code Development Platforms (LCDP), built securely in the cloud infrastructure, to support the transformation of the healthcare system.

Low-Code Development Platforms are being considered by enterprises around the world to deliver rapid software development, continuous delivery, and continuous integration of their application systems. The William Osler Health System is recognized for its adoption of technological innovations for improved patient experience and satisfaction. Its innovations include the use of the Microsoft Cloud for Healthcare platforms; and Microsoft 365 and Power Platform services with embedded Low-Code technology to automate and optimize internal operational processes.

The aim of this master's thesis is to demystify the concept of cloud-based Low-Code Application Development approaches to healthcare software development by using a case study of a healthcare application within the systems being built to support operational processes in the

William Osler Health System. This study contrasts challenges of current internal tools and methods of operations, communication, and application development in the organization, with the potential benefits of using cloud-based Low-Code platforms to drive digital transformation.

Keywords: Cloud-based Healthcare, Application Development, Digital Transformation, Software Development Lifecycle, Software Development Methodology, Low-Code Application Development (LCDP), DevOps, DevSecOps.

List of Figures.

FIGURES

List of Tables.

Acknowledgement

This thesis is written as part of the fulfillment of the Master of Science in eHealth program at McMaster University, Hamilton Canada. As a technology enthusiast, I am always seeking new ways, approaches and innovations that can be leveraged to optimize the operations of Healthcare Systems in Canada - hence the development of this paper.

It is important to recognize the contribution and support received in developing this paper. First, I would like to express my deepest gratitude to my thesis supervisor Dr. Norm Archer who has been very supportive and instrumental throughout the process. Thank you for being my backbone and motivation. Second, I would like to appreciate the Staff and Developers at William Osler Health System, Information Intelligence and Technology Innovation unit for sharing their experience, support, and contribution to this paper.

I would love to also appreciate members of my support system, especially Gord and Valerie Warren and the entire Warren family for giving me a soft landing and for being my new family in Canada. Being an International Student far from family especially during a pandemic comes with its challenges but your presence in my life gave me the strength I needed to get to the finish line.

Finally, I would like to thank the faculty and staff of the eHealth program at McMaster University, Hamilton Canada. Thank you for the knowledge and guidance that has helped kickstart my career in the Digital Health sector. It has been an exciting educational period of learning and gaining new insights on Healthcare Informatics and Digitalization.

List of Abbreviations

1. AD&D – Application Development & Delivery
2. CI/CD – Continuous Integration & Continuous Deployment
3. EHR – Electronic Health Record
4. EMR – Electronic Medical Record
5. HIPAA - Health Insurance Portability and Accountability Act
6. HITRUST - Health Information Trust Alliance: HITRUST Alliance enables vendors to demonstrate compliance to HIPAA requirements based on a standardized framework.
7. LCDP – Low Code Development Platform
8. PHI – Personal Health Information
9. PHIPA – Personal Health Information Protection Act
10. PIPEDA – Personal Information Protection and Electronic Document Act
11. SDLC – Software Development Lifecycle

Table of Contents

CHAPTER 1: INTRODUCTION

Over the years, there have been major improvements in the development of software applications and systems for hospital and patient record and operations management. One such type of widely adopted system is the Health Management Information System. It appears that, as technological innovation advances into cloud-based applications, some factors exist that decelerate such technological adoption in the healthcare industry. Some major concerns are factors of interoperability, scalability and most importantly security of both patient and hospital data. Ensuring that cloud-based application development follows the appropriate security and operational best practices is key to the successful implementation of such systems.

Generically, application development is the process of creating software that performs different tasks and automates business processes (Zoho, 2020). Popular examples of applications developed for healthcare include the Electronic Medical Record (EMR), a computer-based patient medical record often used for clinical practices or hospitals to maintain clinical records of patients. These records typically include data on patient demographics, medical history, biomedical and diagnostic imaging information as well as laboratory results (C.H.I, 2020). Personal Health Records are related solutions that provide a portal for patients to access and manage their own personal health records (Telus, 2020).

The above healthcare applications may seem to be patient-centric. However, studies show that within the time period 1971and 1992, a major highlight feature of EMRs and EHRs was based on optimizing billing and scheduling systems developed to track and manage patient billing details (Evans, 2016). This is but one business function of a hospital system. The digital transformation of relevant internal hospital business processes will be an integral focus of this paper.

Typically, application development follows a series of phases known as the Systems Development Lifecycle (SDLC) for developing an application. This starts from planning or requirements gathering, analysis, designing, building, testing, implementation, and integration/support. However, there are many different approaches, methodologies and frameworks for software development which have been advanced and used over the years. Some of these are listed in Table 1.

Table 1: Eras of Software Development Frameworks (Wikipedia, 2020)

“Era	***Software Development Methodologies, Processes, and Frameworks***
1970s	*- Structured Programming since 1969.*
	- Cap Gemini Software Development Methodology (SDM) since 1974.
1980s	*- Structured Systems Analysis and Design Method (SSADM) from 1980*
	- Information Requirement Analysis/Soft Systems Methodology.
1990s	*- Object-Oriented Programming (OOP) developed in the early 1960s, and became a dominant programming approach during the mid-1990s*
	- Rapid Application Development (RAD), since 1991
	- Dynamic Systems Development Method (DSDM), since 1994
	- Scrum, since 1995
	- Team Software Process, since 1998
	- Rational Unified Process (RUP), maintained by IBM since 1998
	- Extreme Programming, since 1999
2000s	*- Agile Unified Process (AUP) maintained since 2005 by Scott Ambler*
	- Disciplined Agile Delivery (DAD) Supersedes AUP
2010s	*- Scaled Agile Framework (SAFe)*
	- Large-Scale Scrum (LeSS)
	- DevOps”

Currently, the four most common software development methodologies are Agile Development, Waterfall Development, Rapid Application Development and DevOps Deployment. These methodologies each have their pros and cons; however, some of these approaches such as Agile and Waterfall development are time consuming and might take months or even years before application release and integration. With the increased need for faster release of applications to improve and optimize the healthcare system operations and hospital process automation, a new approach to agile software development that supports the security needs of the institution is inevitably needed.

The Agile approach to application development focuses on continuous delivery of the software components through iterations by the software development team (Agile Manifesto, 2001). This is followed by a separate operations team to deploy and integrate the software application with the production system. This approach is usually inefficient in large organizations.

The introduction of DevOps deployment methodology has brought about a collaborative approach involving the Development and Operations teams that had previously worked in silos when using the Agile software methodology. In the DevOps methodology, there is a shift in organizational culture where Operations is integrated with Development in such a manner that it ensures that working software is available to end-users with improved time-to-market, maximized reliability of software release, and improved overall product quality (Crawford, 2021).

Despite the efficiency of DevOps, a major concern that affects the adoption of this methodology is Security (Mohan, 2016). Data breaches in the healthcare sector, either internal or external, have been a cause of concern over the past years and most especially in recent times with the growing adoption of smart devices, IoT (Internet of Things), information systems, and cloud services in the sector. According to research carried out by Seh A. et al (2020), there have been over 3912 established cases of data breaches in the US healthcare sector out of the 6355 incidents reported within the year 2005 – 2019, which is 43.38% more than in other industries (Seh A., 2020). Two categories of data breaches are: Internal data breaches caused by privilege/permission abuse, unauthenticated access, and loss of sensitive data, and external data breaches from external sources like hacking, phishing, malware, ransomware, or spyware (Seh A., 2020). If software development follows the principles of DevOps while neglecting security considerations this can result in additional redevelopment and testing of the software before it can be implemented.

As a result, advanced development approaches are beginning to include Security as an integral component of development. This approach is called DevSecOps, a development methodology that comprises not only Development (Dev) and Operations (Ops) but also Security (Sec). Security, driven by the critical need for patient privacy, is a major consideration for any application development project done for and within the healthcare system, and this is the motivation for including this topic in my thesis.

The soft side of DevSecOps was highlighted by Sánchez-Gordón, M., & Colomo-Palacios, R. (2020) in a systematic review that deals with Security in DevSecOps as a Culture that needs to be cultivated to be successful in the development process. It was categorized into 13 attributes called the 'DevSecOps Culture' which include (Sánchez-Gordón et al, 2020):

1. Collaboration – emphasizing cross-team collaboration with regards to security.
2. Knowledge Sharing – Training of Development, Security and Operations teams.
3. Continuous Feedback – among teams
4. Continuous Improvement Mindset
5. Communication
6. Responsibility – Shared ownership of security considerations among the team.
7. Trust – as a foundation of DevSecOps teams in adopting Security practices.
8. Experimentation – to compare tools for security automation.
9. Leadership – to encourage cultural change & cultivation of the DevSecOps Culture.
10. Commitment & Agreement – of staff to conform to policies.

11. Blameless – avoidance of criticism that leads to negative emotion which can be damaging to the work relation of the security & development team.

12. Hiring new personnel – with understanding of DevSecOps.

13. Transparency – important to establish full visibility between all teams in the model.

In an enterprise like a hospital system, security and privacy must be treated with utmost importance in order to secure and support the privacy of both hospital and patient data. This requirement is legislated and enforced in Ontario by the provincial government through the Personal Health and Information Privacy Act (PHIPA) (MOHLTC., 2005). Including security requirements in the development process involves collaboration among the Security team and the Development and Operations teams in order to map security practices into the DevOps approach, bringing about the formation of the DevSecOps concept.

Aside from the changes described in development methodologies, there have also been some advances in development environments and platforms. Major IT stakeholders like Microsoft and IBM are taking advantage of cloud computing technology and introducing new ways of developing applications in the cloud. Some of these approaches have resulted in the introduction of Cloud-native application development and Low-Code application development platforms (LCDP).

LCDPs are visually intuitive development environments that simplify and speed-up the process of software engineering, even allowing citizen developers and non-programmers to participate more extensively in building software applications (A. Sahay et al, 2020). Enterprise application developers are beginning to adopt these application development platforms for rapid and continuous software delivery.

There is no denying that there is a changing ecosystem in the methods, processes, types of people and platforms for building apps. The future of enterprise software development appears to be driven more and more by the introduction of cloud-native application development and Low-Code application development platforms (Gartner, August 2, 2019). The way forward is by delivering enterprise-grade apps using platforms that support scalability, artificial intelligence, machine learning, cloud computing, and security.

An area of Digital health/eHealth that is often overlooked is the digital transformation and automation of internal hospital management processes involving healthcare teams. The recent COVID-19 crisis has been a reminder that many of our hospital management systems and processes need to be automated, using application development approaches that are agile, fast and scalable for continuous integration and delivery. The current increased demand by hospitals is to develop, deploy, and scale internal cloud-based applications. In addition, these have to be delivered with greater speed, less risk, and reduced cost, through optimized seamless operations and service delivery. Adopting modern cloud-based application development methodologies and serverless technologies is rapidly becoming the way to meet these demands (Accenture, 2018).

This thesis is designed to demonstrate how a secure Low-Code Development Platform that supports the concept of DevSecOps can help healthcare organizations, IT service management and application development teams to reinvent their approaches to software development for digital transformation and delivery. This is a potentially new research area in Canadian healthcare systems. To help address the items addressed above, the following research questions have been formulated:

Problem Statement/Research Question:

RQ1: With the unprecedented demand on healthcare systems to optimize their internal hospital management processes, how can LCDP be securely leveraged to rapidly and optimally scale Digital Transformation in healthcare?

To help gain proper understanding of the research problem, I have included two other research questions that I will use to help guide the course of the study:

RQ2: Why has Low-Code not been considered an appropriate approach for enterprise/healthcare application development? OR What are the concerns around using LCDP to develop applications for the healthcare system?

RQ3: What are the benefits of using Low-Code Development Platforms to develop applications for healthcare system management?

Beginning below, the thesis moves into Chapter 2, the Literature Review to give an overview of the literature on cloud-based Low-Code development platforms, DevSecOps and digital transformation. Chapter 3 will discuss the research methodology used in the thesis along with presenting as an example the Palliative Care Unit Handover App developed at the William Osler Healthcare System using the Microsoft Low-Code development platform (Microsoft Power Apps). Chapter 4 reveals the findings of the study, Chapter 5 discusses the findings in detail and Chapter 6 presents the conclusions of the thesis.

CHAPTER 2: LITERATURE REVIEW

This thematic approach to the literature review will help give a more thorough understanding of some key themes and concepts necessary to properly address our first research question:

RQ1: "**With the unprecedented demand on healthcare systems to optimize their internal hospital management processes, how can LCDP be securely leveraged to rapidly and optimally scale Digital Transformation in healthcare?**

To investigate this research question, I will overview and critically analyze key literature on the topics of Digital Transformation in healthcare, Cloud Computing in healthcare, Low-Code Development, LCDP and DevSecOps. Additionally, I will analyze the limitations and benefits of using LCDP.

The problem of interoperability involves healthcare EHRs (Electronic Health Record systems) applications across different health care institutions and how to better collaborate and exchange health information among the institutions. However, in other relevant areas, methods and platforms are often neglected that would enable better digital transformation of the health industry using techniques that are agile, deliver projects faster, more secure and with less technical difficulty. These are all approaches that could result from some of the benefits of working with Low-Code.

Additionally, while there has been research on software or application development with the DevOps methodology, very few researchers have picked up DevSecOps and how it can be embedded in the Low-Code development platform infrastructure. I will attempt to give some insights on this through my thesis.

2.1. Digital Transformation in Healthcare

Digital transformation in healthcare is the integration of digital technology in areas of the healthcare ecosystem to improve operations, save time and cost and to deliver a value-based model that enhances patient experience (The Enterprisers Project, 2016).

Digital health experts have continuously strived to bring transformation to the healthcare industry through IT. Over the years, healthcare has posted tremendous gains from digital transformation, through the introduction of technological solutions that help deliver quality patient care experience and optimize business efficiency. Some such solutions include Digital Imaging, EHRs, and ePrescription services to mention a few, all of which have been successfully integrated into hospital IT systems.

With high priorities on saving cost and improved security, healthcare IT has focused on the deployment and maintenance of on-premises data centers and software which were assumed to be the right choice for data management and security during previous eras. As the years went by and new technologies were introduced, it became evident that the future of hospital IT applications lies in the Cloud. We are in the transition process of the adoption of cloud computing in healthcare. Cloud-based systems offer tremendous advantages that are more compelling than the previous on-premises hospital infrastructures. The transitions that we went through and are still going through are depicted by Gopal et al in Figure 1.

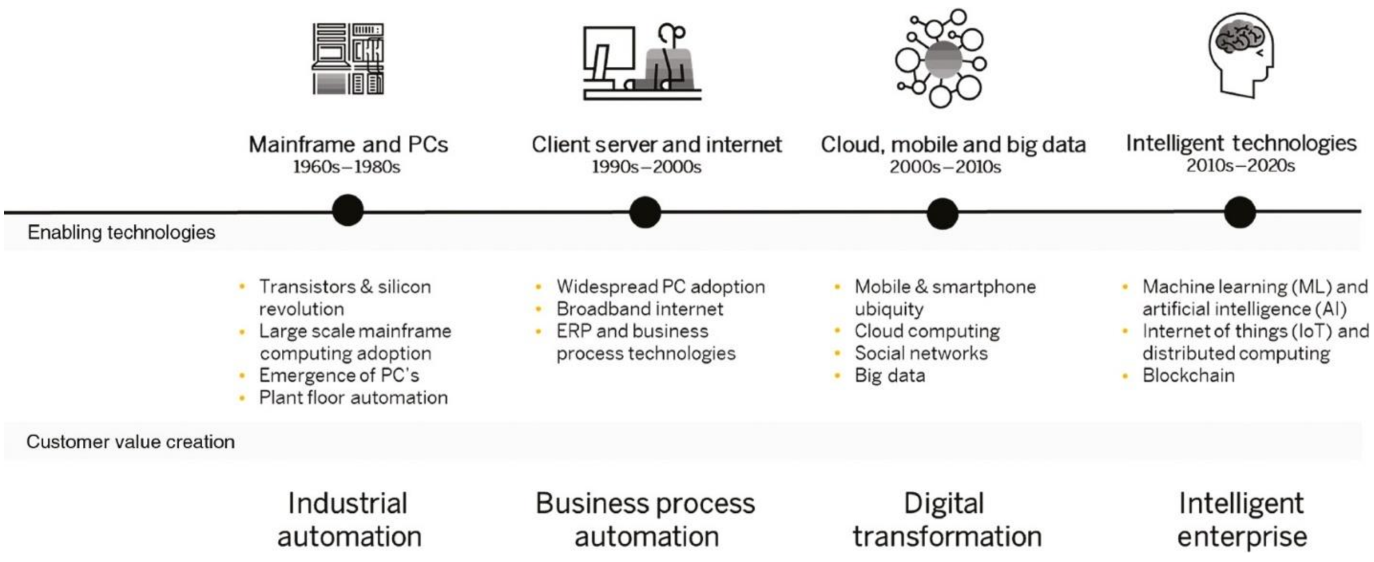


Figure 1: "Evolution of Information Technologies (IT) from the Digital Era into the Intelligence Era." (Gopal, G. et al, 2019)

Gopal et al (2019) mentioned that healthcare organizations need agility to improve patient outcomes and health worker efficiency while at the same time containing cost. Global health leaders have also realized the need to achieve better collaboration in the healthcare system, adequate data access, and proper predictive analytics and insight to achieve greater value from the data retrieved (O.H.A, 2020). All these can be realized through digital transformation and embedded intelligent systems within the healthcare business process.

Throughout the study by Gopal et al (2019), there is a consistent message showing that digital platforms on cloud systems lead to achieving a truly intelligent healthcare enterprise faster and with less risk (Gopal, 2019). Although some major players in health IT are still skeptical about cloud-based systems in healthcare (Davis, (2019) and Eddy (2019)), traditional on-premises architectures or platforms are no longer adequate for delivering software and applications for healthcare at the speed, cost and levels of security necessary for the current technological era (2020s). This is an era where big data, machine learning and artificial intelligence can leverage the power of the cloud system.

Figure 1 illustrates the expected evolution of Information Technology categorized into four eras; industrial automation, business process automation, digital transformation, and the intelligent enterprise. Industries have continually progressed through these technological eras. The healthcare system is not left out, but it is evolving at a slower pace, which we can attribute to the precautious nature of the healthcare industry. That is why the healthcare industry is just beginning to settle into the cloud, mobile and big data era.

2.2. Cloud System/Computing in Healthcare

Peter and Grance (2010) defined cloud computing as a model that enables access to an on-demand pool of computer system resources that can be rapidly provisioned, with reduced IT management and minimal direct user management. Basically, cloud computing is the modern way of delivering IT resources, services, and infrastructure (like servers, storage, databases) over the Internet, using fast deployment methods.

Currently, advances in hospital management and operational intelligence leverage the power of cloud computing by applying artificial intelligence and machine learning to big data produced by hospital systems and stored in the cloud (Greco, L., 2020). But there are challenges to the integration of advanced intelligent technologies like the Internet of Things (IoT), Artificial Intelligence (AI), Machine Learning (ML) and Big Data in healthcare. These problems include data storage and management, security, privacy, and exchange of data between physical devices and cloud computing addresses, shown as the Intelligent Enterprise era in Figure 1. These advanced systems, in addition to cloud computing, hold additional promise to optimizing the business process and the development of a fully digitized healthcare sector.

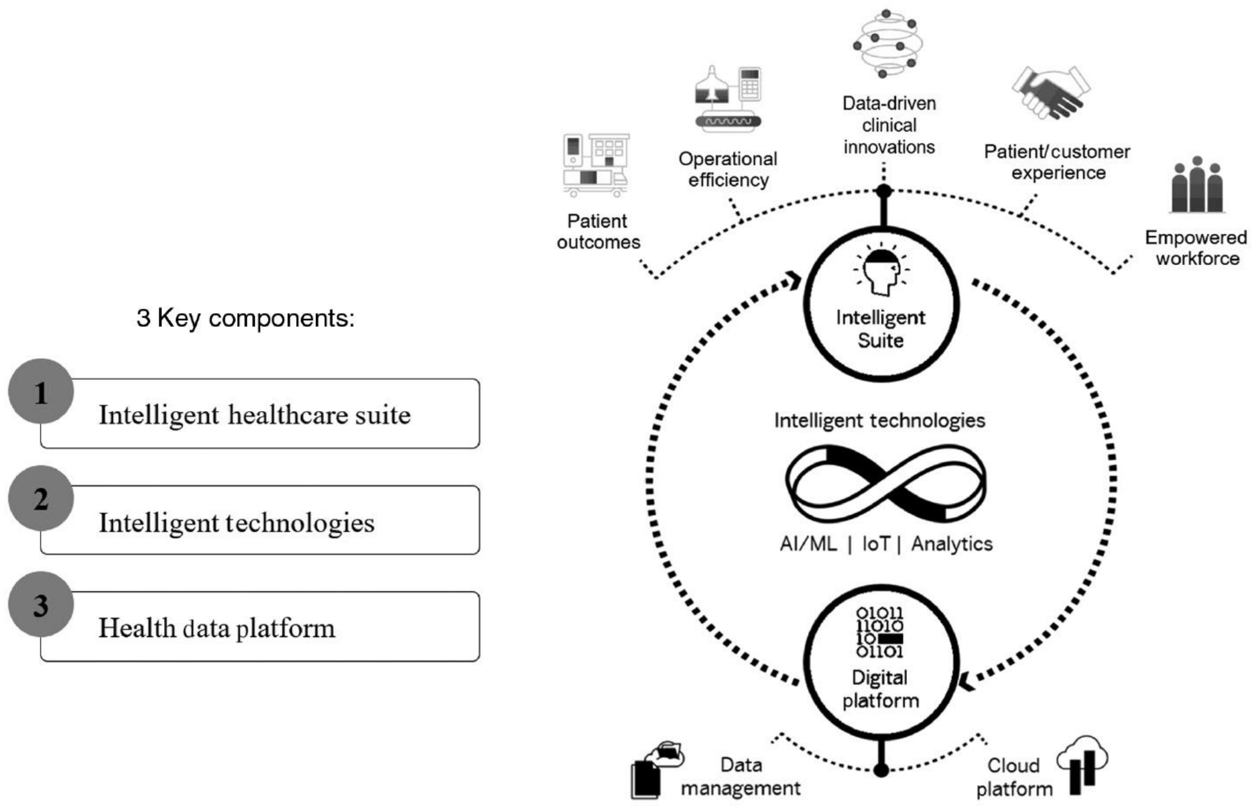


Figure 2: "The Intelligent Healthcare Enterprise framework." (Gopal et al., 2019)

Figure 2 demonstrates the intelligent healthcare framework highlighted by Gopal et al. (2019) as a system with three layers or key components where data management and cloud platforms are the foundation of the framework for an Intelligent Healthcare Enterprise. This provides a digital platform for advanced intelligent technologies like Artificial Intelligence, Internet of Things (IoT) and Data Analytics. These in turn facilitate the creation of intelligent healthcare suites of products that promote patient outcomes, operational efficiency, data-driven clinical innovation, patient customer experience and an empowered workforce. This demonstrates how much the digital transformation of the healthcare ecosystem depends on cloud computing.

Some advantages of cloud computing that have been demonstrated for healthcare include easier storage and archiving, use of patient records and medical imaging to build predictive models for machine learning, savings on storage costs, increased pace of innovation cycle releases, augmented telemedicine services and improved collaboration, to mention a few (Gopal, 2019).

According to recent research, cloud computing is gradually evolving into a decentralized paradigm called Fog Computing, a recent development coined by *Cisco* in 2014 that consists of edge nodes directly connected to physical devices (Sakovich, 2020). But no matter how it is configured, cloud computing is still very relevant to research around leveraging LCDPs for optimized app delivery.

2.3. Low-Code Development

Conceived in 2014 by *Forrester Cambridge Research* (Sanchis, 2019), the term "Low-Code" was discussed in a report entitled "*New Development Platforms Emerge for Customer-Facing Applications*." as a modern alternative to application development for faster and continuous delivery. The Low-Code platform enables rapid delivery of business applications with minimal coding, setup, and investment requirements for app deployment.

In agreement with Forrester (2014), Revell et al. (2020) also defined Low-Code as a software development approach that enables the fast delivery of software applications with minimal hand-coding using visual modelling in a drag-and-drop graphically intuitive interface for configuring applications (Revell, 2020). According to Revell et al. (2019), Low-Code is used as a noun just like you would describe the programming languages 'Python', 'Java' or 'C#'. It can also be used as a verb because it describes a method of software development that involves the process of integrating visual blocks of already existing code into an interface and workflow to create a working application.

In the Low-Code development approach, rather than having developers spend countless hours of time writing thousands of lines of code requiring complexity related to syntax and semantics, it allows users to build complete end-to-end applications with modern interfaces, integration, data, and logic with less complex levels of effort, investment, and time. This enables skilled developers to work faster and smarter without having to do repetitive coding. This is an integral motivation leading to predictions by Gartner that Low-Code will be used for over 65% of the application development market by 2024 (Southern, 2020).

We should also note that Low-Code does not mean the absolute absence of coding in software development, but it only requires minimal code writing and less complexity than the traditional application development approaches. Some Low-Code platforms require some knowledge of functions and expressions (for example those used within the Microsoft Excel application) to configure elements of the application, but some require additional coding for specific situations. However, note that there is a concept often mistaken for "Low-Code" which is called "No-Code". This is a platform for developers who do not know any programming languages, providing everything the user would need to build an app without much customization effort. The downside to No-Code is that its tools are difficult to tweak and customize and do not have any unique functionality. An example of No-Code solutions would be the blogging and ecommerce website platforms used to launch new websites in times as short as hours to minutes.

Findings from the literature reviewed under the Low-Code theme (Richardson, 2014 & 2016) show great growth and prospects for the Low-Code market in coming years although traditional development methods of building applications with core programming languages are still prevalent today (Rymer, 2019). In an era of remote work and rapid development, traditional development methods are no longer adequate to meet the pace of software delivery needed to achieve patient satisfaction in the healthcare sector. Low-Code is fast gaining momentum in many industries. Overall, Low-Code development is acting as a form of digital transformation accelerator in the healthcare system.

2.4. Low-Code Development Platforms (LCDP)

Low-Code development activities are carried out in Low-Code Development Platforms (LCDPs). According to the Forrester definition, LCDPs are: *"Products and/or cloud services for application development that employ visual, declarative techniques instead of programming and are available to customers at low or no cost in money and training time to begin, with costs rising in proportion of the business value of the platforms"* (Rymer, 2017, p. 4).

On a basic level, LCDPs are cloud-based software development platforms delivered through a Platform-as-a-Service (PaaS) model that allows users to build fully functional and operational software applications for web and mobile devices by interacting with intuitive graphical interfaces, visual diagrams, and declarative languages (Tisi, 2019).

LCDPs leverage advancements in visual programming and cloud infrastructure. Major PaaS vendors (like Microsoft, Salesforce, OutSystems, etc.) (Rymer, 2019) are integrating LCDP into their suites of products and solutions to provide a managed environment for developers. The literature (Richardson, 2016) shows that LCDPs have been successful in the development of applications for many different domain-specific segments such as: General-purpose Low-Code Platform Apps, Database Apps, Process Apps, Mobile Apps and Request Handling Apps. A more recent wave of innovation on Low-Code platforms has produced apps that include devices for Internet of Things (IoT) technology (Sanchis, 2019)

Although there are several different vendors that service the LCDP market, we have found that Microsoft is one vendor that services all domain-specific segments of applications developed with Low-Code, which makes it an ideal product for the healthcare market. Microsoft has a product called the Power Platform, an intelligent tool built on the Microsoft Cloud Platform and Azure infrastructure. It is interoperable with the Microsoft 365 suite of applications and William Osler Health System's current messaging, storage and collaboration platform. It is therefore a good choice of platform to transition to Low-Code development. According to Itvision (2020), 97% of the Fortune 500 companies use this power platform, which is a proof of the trust companies have in the Microsoft Power Platform.

2.5. Limitations/Concerns of LCDPs

As with any new technology, there are a few concerns or limitations with LCDPs. These have been highlighted by Tisi M. (2019) who considered three main limitations to the use of LCDP as:

- Scalability: Previously, Low-Code was popular for small applications, which was considered a limitation to its adoption. However, in recent times, Low-Code has evolved and is being used in large scale and mission-critical enterprise application development.
- Fragmentation: Different vendors have proposed their own Low-Code paradigms to be used along with their specific programming models.
- Software-Only System: Some citizen developers are experts in other domains without a solid understanding of software engineering and may not have adequate experience when it comes to making sound decisions regarding how computer systems should work.

Additionally, a survey carried out by OutSystems in 2019 (OutSystems, 2019) to determine why some organizations are still not using Low-Code shows that organizations were mostly concerned about the lack of knowledge about Low-Code platforms in addition to concerns about lock-in, scalability, flexibility and security. Figure 3 shows a chart of their survey results.

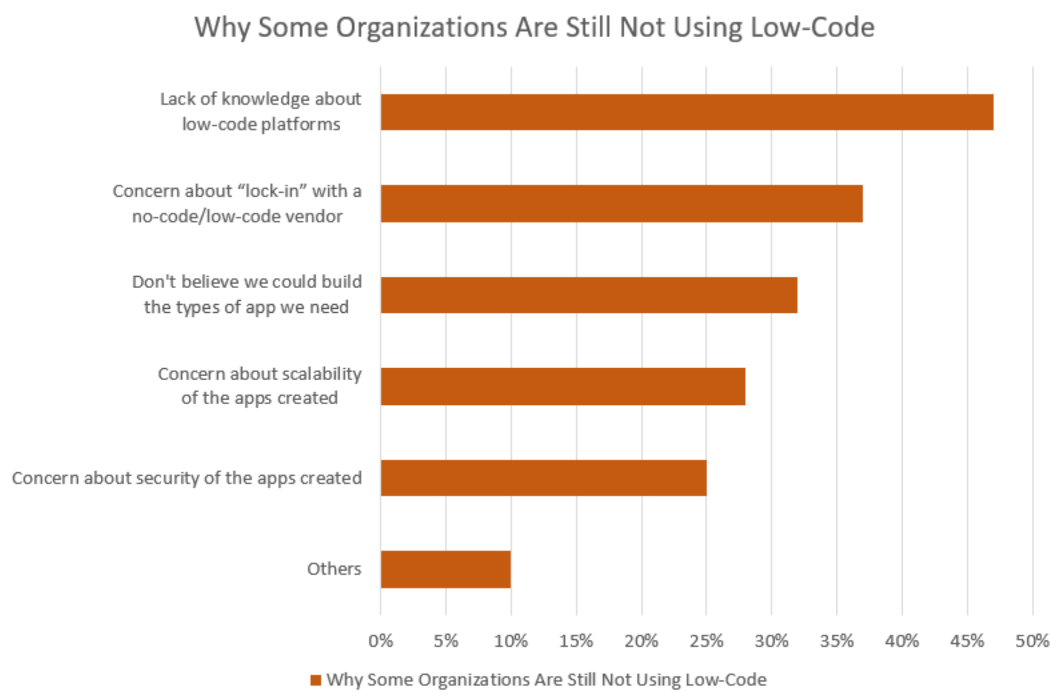


Figure 3: OutSystems Survey Reasons for not Considering Low-Code Platforms (Source: OutSystems)

The reasons stated above reinforce the need to demystify the use of Low-Code as an appropriate option for application development in healthcare systems.

2.6. Benefits of Using LCDP

Making the choice to build enterprise-critical applications with a Low-Code platform comes with certain perks along with the means to create complex and efficient applications rapidly. Some of the main benefits are (Ness & Sanchis, 2019):

- Speed: Low-Code accelerates development and deployment time
- Agility: Low-Code enables enterprises to pivot effectively to new digital strategies and initiatives caused by industry changes and patient or customer needs.
- Innovation: Low-Code encourages and facilitates innovation by allowing the building and release of innovative solutions that digitize business processes and deliver future-proof applications that are secure.
- Increased Development Talent: With Low-Code, citizen developers can contribute to the efforts of the development team.
- Automated Governance: Low-Code has built-in automated governance which enables for appropriate monitoring and control of development projects within the organization, thereby creating a secure apps environment.
- Cost Reduction: In terms of time and level of complexity, Low-Code reduces the cost that would be incurred in developing apps by professional developers.
- Easy Maintenance: Since Low-Code offers little code, there is a relatively reduced demand for maintenance.
- Reduced Complexity: With Low-Code the applications are built in a graphical environment that reduces the complexity of developing purely with code.

Developers are also beginning to feel less threatened that a low-code platform might replace their jobs and have come to realize that it provides some direct benefits to developers. It enhances job security for developers as it enables them to build apps faster with fewer defects, thereby speeding up the release lifecycle. It also reduces repetitive coding and enables developers to focus their skills where they are most needed.

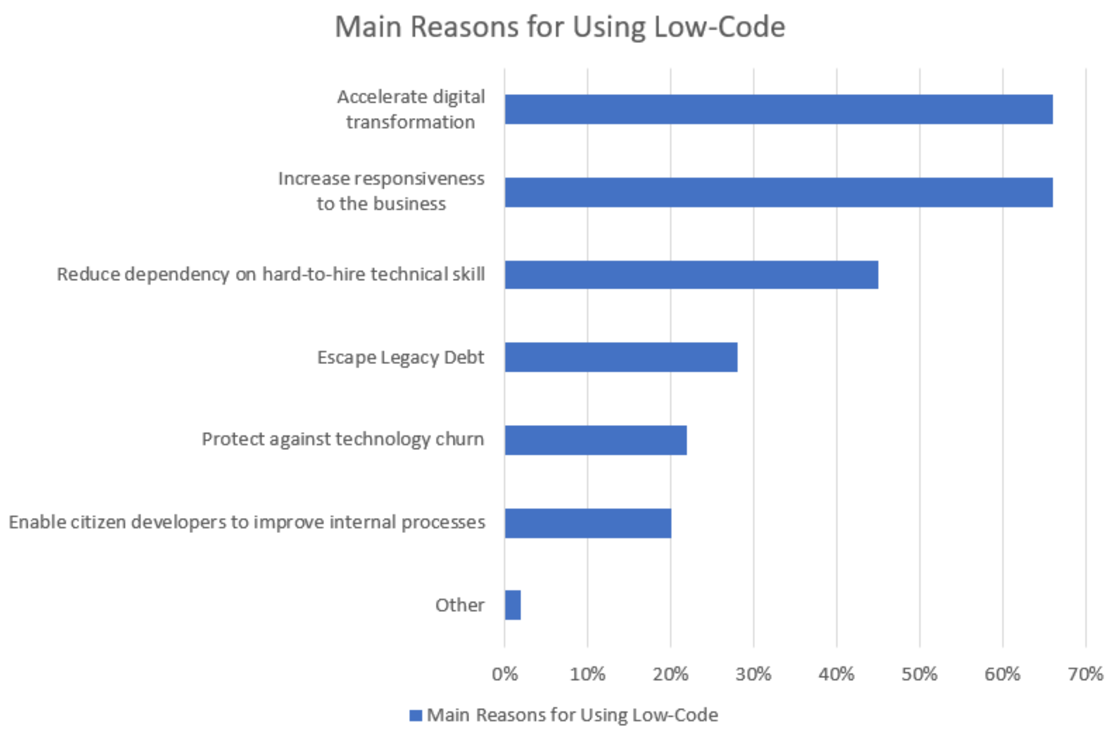


Figure 4: OutSystems Survey Results About Reasons for Using Low-Code Platforms (Source: OutSystems)

Figure 4 shows results from the OutSystems survey on the main reasons why organizations chose to use Low-Code. The top two reasons were to accelerate digital transformation and to increase responsiveness of the business. These were along with other reasons which included to reduce dependency on core hard-to-hire coders, reduce the need to support legacy systems, and to close the technology skill gap. These highlighted needs for using Low-Code would also likely hold true for Healthcare Systems which are currently in most need to increase responsiveness to business-critical needs of the sector, especially in these times of pandemic.

2.7. Low-Code Supports DevOps/DevSecOps

One of the goals of DevOps is to bridge the gap between IT and Operations. Adopting DevOps within an organization reduces the need for manual tasks that extend developer time, leading to delivery backlogs. In DevOps, teams collaborate to accelerate delivery of digital applications (see Fig5).

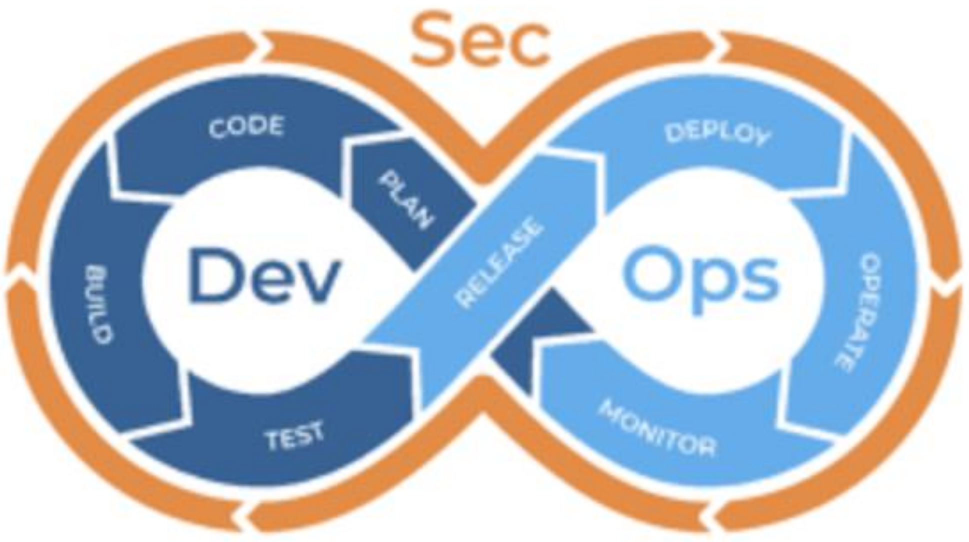


Figure 5: DevOps with Security (Subramani S., (2019). DevOps with Security [Image]. FlexMind. https://www.flexmind.co/azure/practicing-devsecops-with-azure-devops/)

DevSecOps is a recent concept that integrates security into all stages of the DevOps methodology. Fitzgerald and Stol (Neely, 2013) described Continuous Security as "*transforming security from being treated as just another non-functional requirement to a key concern throughout all phases of the development life cycle and even post deployment, supported by a smart and lightweight approach to identifying security vulnerabilities.".* DevSecOps is a journey to continuous security that optimizes development and operations methodology to an advanced level of security consciousness (Ahmed, 2019). Just as DevOps strives for effective collaboration between development and operations, as illustrated in Figure 5 DevSecOps includes the Security team within this collaboration.

Low-Code also supports DevSecOps, by improving the bottom line and increasing the efficiency of DevOps in the following ways (Anderson, 2020):

- Low-Code helps to alleviate talent shortages and skills gaps by broadening the application development space to non-technical professionals who are able to support the organization's digital transformation mandate. This way, instead of having a traditional application development team that is responsible for the delivery of applications in the face of an ever-increasing digitization demand, organizations can free up their professional coders from routine programming tasks and strategically redeploy them to tasks or projects critical to core business processes.
- Low-Code facilitates rapid application development. It speeds up the entire build and deployment process thereby reducing deliverable timelines from months to weeks or days, thereby accelerating app delivery.
- Low-code platforms automate and accelerate the deployment process. Low-Code streamlines application deployment since it integrates more seamlessly with the business environment, thereby minimizing disruption and irregularities.
- Most enterprise LCDPs integrate security right from the starting point.
- Most LCDPs also provide analytics tools, allowing the development team to measure app performance, collect feedback, and rapidly identify potential problems.

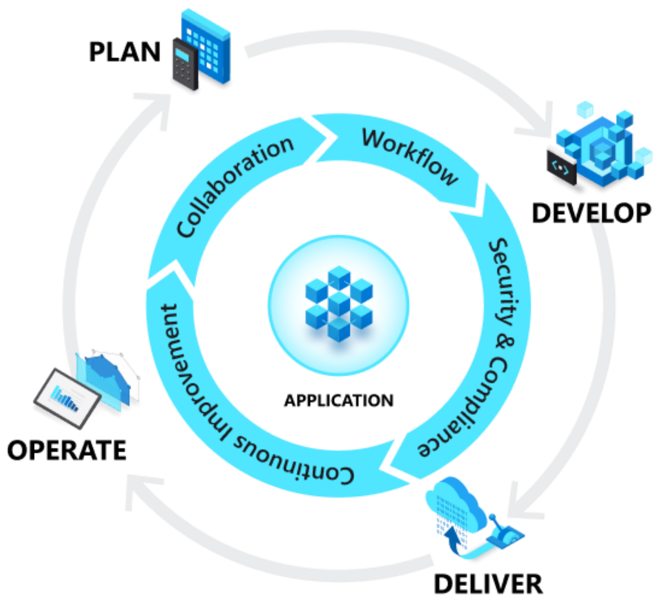


Figure 6: DevSecOps and the Application Lifecycle (Microsoft, 2020)

The DevOps tools needed to fully embrace DevOps practice are steadily increasing in number (Linthicum, 2020). Over the course of time, organizations often acquire disparate tools to support DevOps and they end up with a complex collection of solutions for version control, code validation, test automation, automated deployment etc. (Linthicum, 2020). This increases the level of complexity, management, IT budgets and wastes time.

To reduce this complexity and waste and to maximize efficiency of the DevOps process, IT teams are starting to adopt advanced LCDPs (Dan, 2018). These platforms have the capability for collaborative development, version control, build validation and testing. They provide users with environments that encompass the entire application development lifecycle including build, testing, deployment, monitoring, and maintenance while including considerations of security, governance, and compliance at every level. Having all these capabilities in one platform solution greatly improves the output and speed of the DevOps team.

Gartner (Fusion Teams, 2019) refers to the model in Figure 5 as a “Fusion Team”. This refers to a multidisciplinary digital business team that brings together IT and business employees to develop digital solutions critical for successful digital transformations. In this phenomenon Gartner (2019) points out that business users, who understand the requirements of what they need, can participate in the development process either by building apps themselves or by working alongside developers who leverage Low-Code. Developers help in the process by doing the hard parts of developing code to provide access to the Application Programming Interfaces (APIs), writing custom controls when necessary, and dealing with more complicated business logic but not having to do a project that requires complex programming to build user-facing front-end and back-end applications. Developers rely on the DevOps tools, source controls, and lifecycle management services that are provided and supported in the operations platform. If IT staff trusts these services, they can grant user access to them. In this way IT staff governs, monitors, and empowers citizen developers and business units in the organization. Of course, citizen developers must register apps they are working on within this environment and know where to go for advice and support without heavy involvement of IT staff in every single project.

Figuring out how to make the fusion team (otherwise called the ‘power team’) work together fundamentally changes the economics and the pace of delivering solutions for the enterprise. Their approach can deliver critical lines of business applications to the healthcare system efficiently in a very short period of time while ensuring that the development output stays in compliance (Gartner, 2019). This is part of the main crux of my research. Healthcare systems that are able to strategically leverage Low-Code solutions in combination with modern DevOps practices are likely to sustain their growth and succeed.

CHAPTER 3: RESEARCH METHODOLOGY

Things to consider when deciding on a research design include the research problem, the current research on the topic of interest, and the required data type. At the time of doing this research, there were very few publications on research topics associated with the use of low-code in the healthcare system in Canada. For this reason, the research methodology was based on a case study of the use of low-code at the Palliative Care Unit of the William Osler Health System.

I developed an app using the Low-Code development approach to digitally transform the patient handover process from the current process in use at the Palliative Care Unit at William Osler Health System. This application involves workflow process automation to automate and optimize some elements of the patient handover process. This was used to demonstrate that the modern approach to building, designing, and maintaining software in the cloud increases the agility of the software development team to release applications faster, more reliably and securely.

3.1. Background

To briefly explain the setting of the research, the Palliative Care Unit formerly managed its patient handover process as elaborated below.

The Palliative Care Unit previously used Excel spreadsheets to record patient information to be transferred between physicians. These Excel files were then stored on physical servers within the organization's data center. Since this system was developed, demand increased and processes became more complex, so it was becoming hard to track records in the files, and there were increased chances of error. With no centralized data repository, it was hard to keep track of the most current version of data files, there was no proper information security, information could go missing, and users were required to be physically present in the hospital to have access to the particular drive where the Excel sheets were stored. Realizing this was not the best way to operate increased the need for the digital transformation of the Palliative Care Unit, resulting in the birth of the PCU Handover app.

William Osler Health System (WOHS) is a public hospital system that serves residents of the Western Greater Toronto Area with locations in Brampton, Etobicoke, Peel region and surrounding communities within the Central West Local Health Integration Network - LHIN (William Osler Health System, 2021). The system has three major operating centers: the Brampton Civic Hospital, Peel Memorial Center for Integrated Health and Wellness, and the Etobicoke General Hospital. Due to the Covid-19 pandemic, the ever-busy emergency department and other units of care at WOHS centres became even busier with patients in need of urgent care, making it among the busiest hospitals in the province.

The WOHS vision statement is to create "*Patient-inspired health care without boundaries" along with its mission to deliver "innovative health care delivered with compassion*" (William Osler Health System, 2019). They have a strategic plan for 2019 – 2024 "Going Beyond for Healthier Communities" in which the goal is to implement innovative, collaborative, and sustainable change that would transform and improve the way they deliver care and how patients engage with the health system, thereby creating a sustainable future for healthcare at WOHS and the communities it serves (William Osler Health System, 2019). After launching the strategic plan in 2019, WOHS joined the 'Brampton Etobicoke Ontario Health Team' set up by the Ontario Government to enable healthcare providers to work in coordination to ease the transition of patients between systems and to strengthen local services.

The global pandemic in 2020 caused an update in the strategic plan to a corresponding 'Strategic GO Plan' as shown in Figure 7, which focuses more on prioritizing projects that will have broad organizational impact and are foundational to the System's long-term success during and after the global pandemic. A major strategic project under the Digital Transformation initiative is the implementation of communication and collaboration platforms to optimize organizational effectiveness. This aims to optimize value through efficient and effective use of resources to deliver healthcare that the patients and community can rely on.

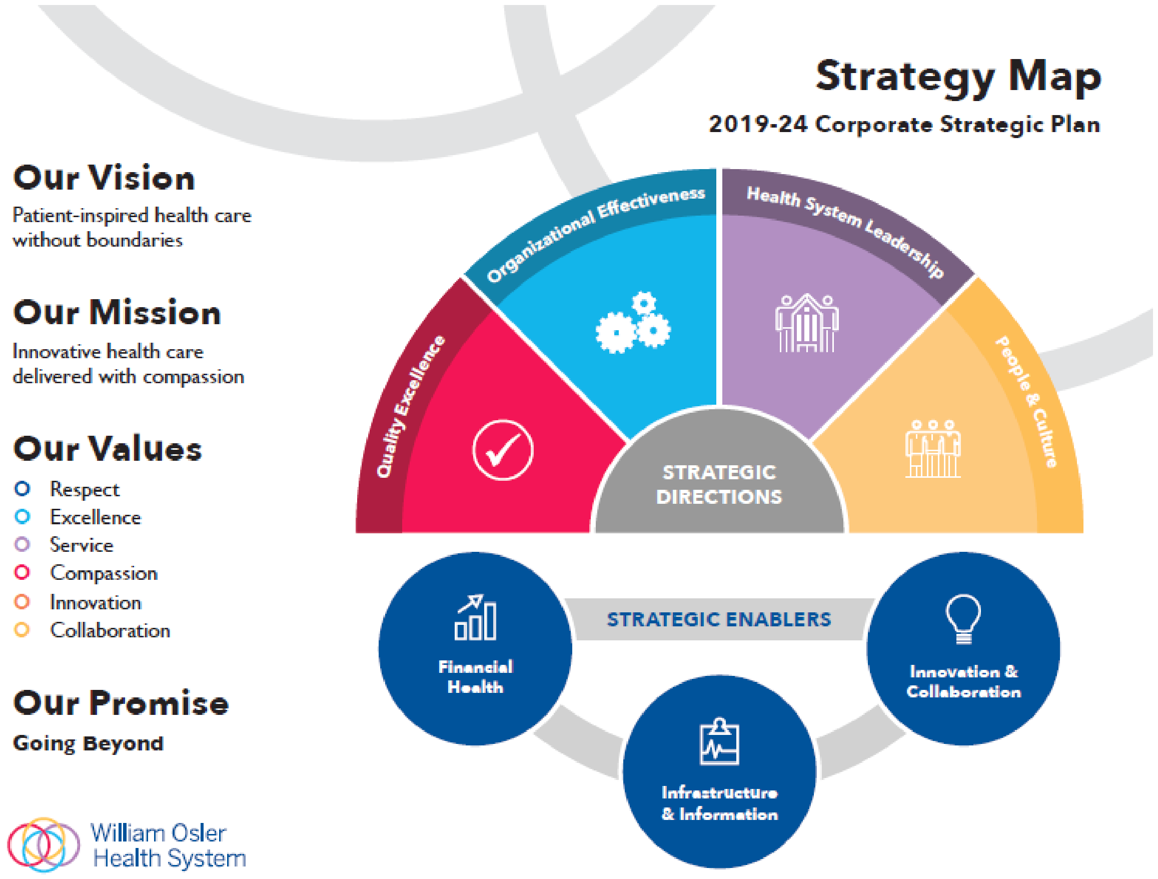

Strategy Map
2019-24 Corporate Strategic Plan
Our Vision
Patient-inspired health care without boundaries
Our Mission
Innovative health care delivered with compassion
Our Values
Respect
Excellence
Service
Compassion
Innovation
Collaboration
Our Promise
Going Beyond
William Osler Health System
Quality Excellence
Organizational Effectiveness
Health System Leadership
People & Culture
STRATEGIC DIRECTIONS
STRATEGIC ENABLERS
Financial Health
Infrastructure & Information
Innovation & Collaboration

Figure 7: An Outline Illustrating the William Osler 'Strategic GO Plan'. (Source: William Osler Health System, 2019)

As part of the transformation process, the Patient Handover App for the Palliative Care Unit at WOHS is to be transformed into an electronic care coordination, health record and collaboration tool for keeping track of patient symptoms for those with terminal illness and at end-of-life phase. This ensures a safe handover of patient records from one physician to another as they focus on improving the quality of life of their patients, providing the best comfort at end-of-life and possibly prolonging life of the patients. This will be built with the Microsoft 365 suite of products for collaboration and using the Microsoft Power Platform for low-code development. Leveraging on the connective nature of this platform, the seamless flow of information, components, APIs and connectors with security will be better achieved.

3.1.1. Low-Code Platform Selection

To provide more detail, the platform at WOHS is the Microsoft Cloud and Low-Code Development Platform (from now on, abbreviated as the Microsoft LCDP). In the Forrester Wave report (Rymer, 2019) illustrated in Figure 8, Microsoft was described as a leader among 13 other significant platform providers. All 13 participating providers offer designs that can serve the needs of Application Developers and Delivery professionals, providing declarative development approaches, low-cost-of-entry models, ability to serve large enterprises and support many business use-cases (Rymer, 2019).

Forrester's evaluation was based on 28 criteria grouped into 3 categories: product and customer engagement strategy, current offering, and market presence (Rymer, 2019).

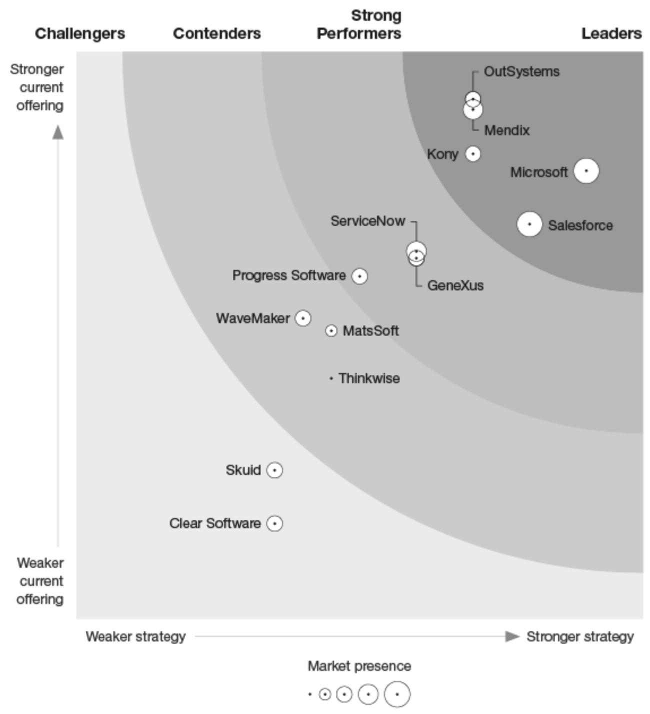


Figure 8: Forrester Wave™ Research: Low-Code Development Platforms for AD&D Professionals Scorecard (2019).

More recently Gartner Inc., another global research and advisory firm published a report in September 2020 on *"Magic Quadrant for Enterprise Low-Code Application Platforms." This is i*llustrated in Figure 9. In the report, the evaluation criteria were categorized based on the ability to execute and the completeness of vision. The Microsoft LCDP was found to be one of the top leaders in the industry with strengths in product strategy, innovation, API (Application Programming Interface) and integration services (Vincent, 2020). The fact that the Microsoft LCDP is one of only four vendors to be top-ranked by both Forrester and Gartner makes it a good LCDP choice.

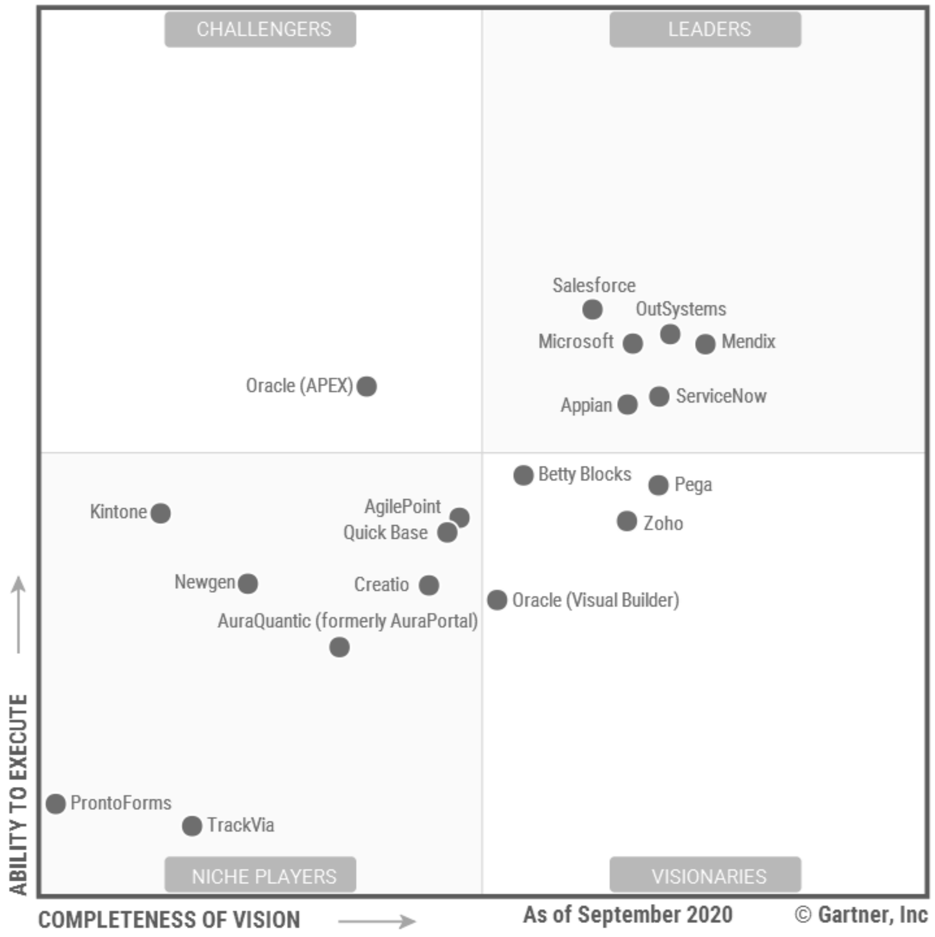


Figure 9: Gartner Research: Magic Quadrant for Enterprise Low-Code Application Platforms (September 2020)

The Microsoft platform offers powerful features to rapidly develop enterprise-grade business applications for the healthcare system by providing web and mobile user experiences along with digital process automation, Artificial Intelligence/Machine Learning (AI/ML) capabilities, a large catalog of integration connectors and a Low-Code language with very close affinity to Microsoft Excel formulae/expressions known as 'Power FX' (Krill P., 2021). Although the platform may require some form of product training for the AD&D (Application Development & Delivery) professional, it is regarded as a leading choice among available low-code development platforms.

3.1.2 Privacy, Security & Compliance Considerations

Privacy, security, and compliance were major considerations in the LCDP platform selection process at WOHS. It is important that healthcare organizations work with cloud solutions and vendors that are compliant to appropriate healthcare regulatory bodies such as PHIPA (Personal Health Information Protection Act) and PIPEDA (Personal Information Protection and Electronic Document Act) in Canada, and HIPAA (Health Insurance Portability and Accountability Act) in the United States. To this end the Microsoft LCDP also proved to be the best provider for the digital transformation project at WOHS. The Microsoft LCDP is PIPEDA & HIPAA-compliant (Robert, 2021) making it even more useful for digital transformation developments, by following its proper security, privacy, and compliance standards throughout the development process.

As a cloud-based approach, the concept of tenancy and environment (shown in figure 10) creates security boundaries that are used to wrap all users, apps, data, and connections. Environments can be set up within the low-code development platform specifically for different business units (including test or production) while maintaining control over user permissions and access within those environments. This assures that the applications are built on the concept of trust at every level due to supporting control and governance.

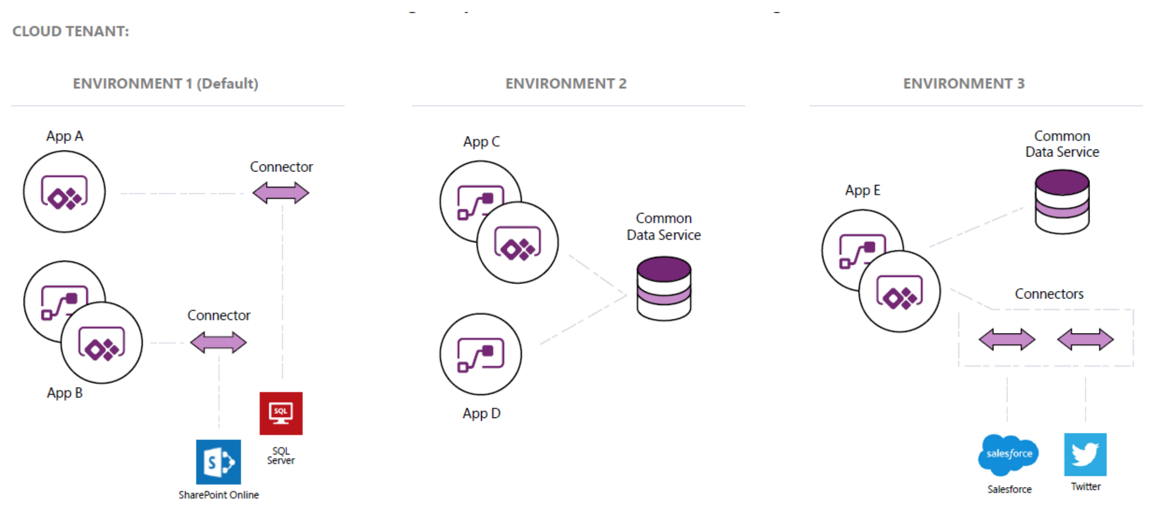


Figure 10: Microsoft Power Platform Low-Code Tenancy and Environment Concept (Source: Microsoft Inc.)

Additionally, the Microsoft LCDP has layers of security that protect data privacy and integrity by implementing user-based and role-based security, through policy governance and support for multi-factor authentication within an isolated secure development environment. This, together with the hospital's security protocols that also embrace the full lifecycle of protected health information (PHI) gives an extra assurance of security compliance.

3.1.3. Microsoft Power Platform

The Microsoft Power Platform LCDP is a suite of business applications that enable rapid application development by providing data insights and easy solutions to business problems. The Microsoft Power Platform is a collective term for four products (Darshan, 2021) shown in Figure 11 and consisting of:

- Microsoft Power Apps: To build and launch web and mobile applications using prebuilt code components with drag-and-drop capability for rapid deployment and continuous improvement.

- Power BI: Business Intelligence tool for data and business analytics applications to find and share meaningful insight with data visualizations and built-in AI capabilities along with integration to prebuilt connectors.
- Power Automate: To build time-saving workflows for process automation with seamless integration to a wide variety of prebuilt connectors.
- Power Virtual Agent: To build powerful intelligent conversational chatbots without the need for data scientist intelligence support.

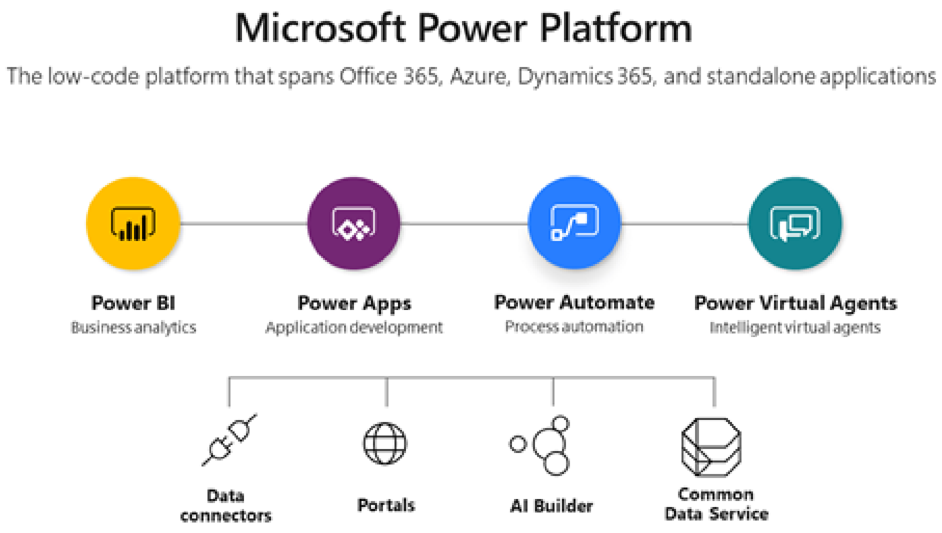


Figure 11: Microsoft Power Platform (Source: Microsoft online)

Prior to the introduction of the Power Platform, creating an end-to-end application or solution within the hospital system required core professional developers, a team of data scientists, disparate technologies, and a lengthy completion time for the project while following the traditional application development method. The Power Platform transformed that process to a more rapid, intuitive, collaborative, and secure solution without the need for complex coding. Figure 12 describes the trust and security layers provided by the platform which is built on the concept of trust at every level and supports control and governance as well as its integration with already existing security measures, identity management system and Azure Active Directory.

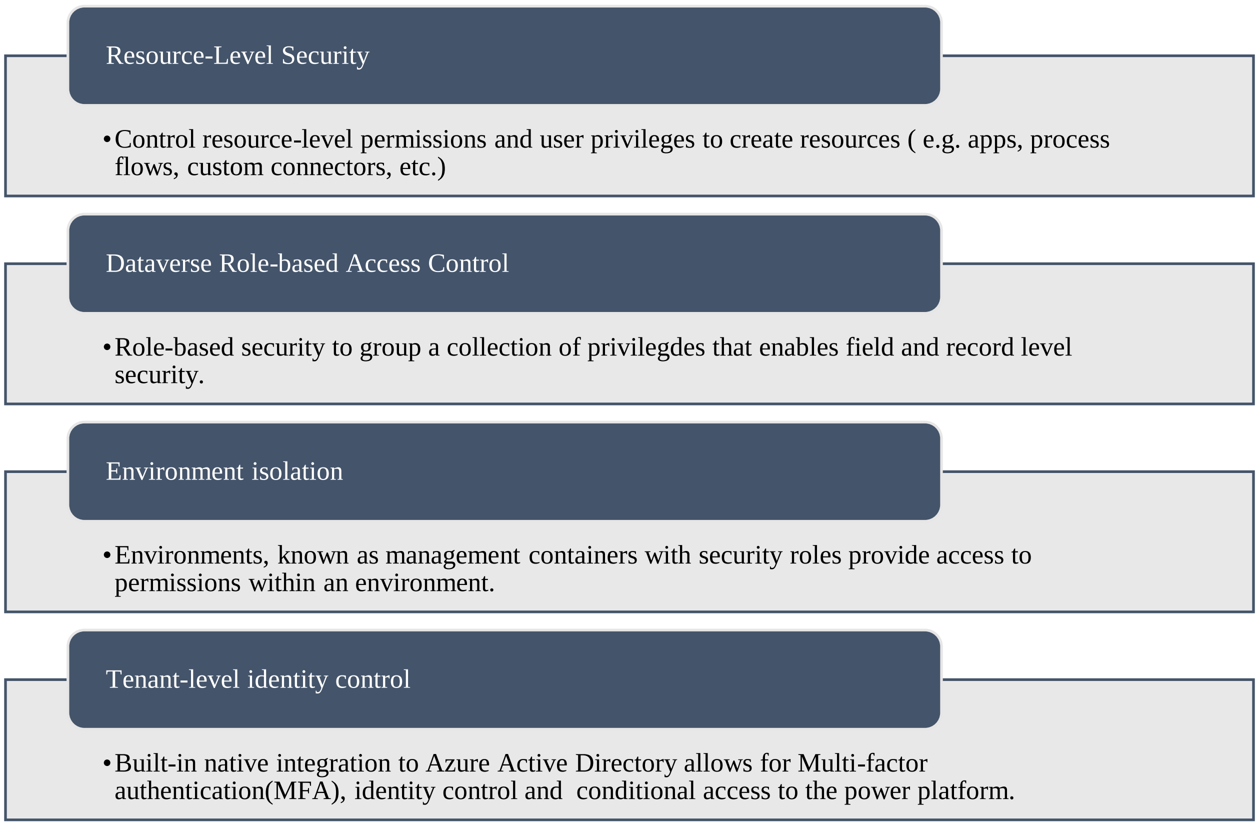

Resource-Level Security
•Control resource-level permissions and user privileges to create resources (e.g. apps, process flows, custom connectors, etc.)
Dataverse Role-based Access Control
•Role-based security to group a collection of privilegdes that enables field and record level security.
Environment isolation
•Environments, known as management containers with security roles provide access to permissions within an environment.
Tenant-level identity control
•Built-in native integration to Azure Active Directory allows for Multi-factor authentication(MFA), identity control and conditional access to the power platform.

Figure 12: Microsoft Power Platform – Trust & Security Layers (Source: Microsoft Inc.)

3.2. Comparison as a Research Method

To answer the research question, I carried out a comparative analysis of the traditional and low-code approach to application development using different criteria relevant to answering the research question. This method was chosen to distinctively portray the benefits of leveraging low-code development approaches to digital transformation in the healthcare system.

3.2.1. Criteria for comparison

To compare the Low-Code and the traditional development approaches, I applied principles from the Project Management Institute (PMI) book, *A Guide to the Project Management*

Body of Knowledge. (PMBOK). These principles have been adopted by many organizations around the world including the Institute of Electrical and Electronics Engineers (IEEE) as the project management standard to keep project management concepts and expectations consistent. These PMI knowledge areas have been translated based on the nature of this thesis project into criteria for comparison between the two development approaches as below:

1. Scope management: This looks at how well the two approaches manage stakeholder needs and requirements as well as managing changes to the scope baseline.
2. Time management: This looks at estimates and the total amount of development and testing time it should take both approaches to complete the development of the app.
3. Cost management: This compares the approximate monetary value that both approaches might incur.
4. Human Resource management: This compares the human resources that would be required to develop an app using either of the two approaches.
5. Communication Management: This determines how well teams collaborate in developing and distributing the app within either context.
6. Risk Management: How each approach addresses privacy, security, and compliance.

Additionally, I will be discussing the following criteria for both the Low-Code and Traditional approaches:

- Ease of maintenance: The maintenance response time for each development approach.
- Mobility: How the two development approaches aid mobility. How well do they leverage the cloud platform to aid mobility?
- Upgradability: Version control and app upgrades are done in both the Low-Code and traditional development approach. Which is more efficient?

3.2.2. Selection of Developers/Respondents

The collection of data and results of the comparison was documented by two developers from the Information Intelligence and Technology Innovation (IITI) unit at WOHS:

- I served as an IT developer proficient in Low-Code development, working with the Microsoft Cloud and Low-Code platforms as a Business Intelligence Consultant with the IITI unit.

- A Professional Application Developer proficient in programming languages such as C# for backend, Cshtml (razor syntax), HTML, CSS, and JavaScript estimated the development of an equivalent app using native programming codes. This was done by Gurinder Bassi, an Application Developer with the IITI Unit.

3.2.3. PCU project requirements

For the purpose of the thesis, I developed the Palliative Care Unit Handover Application with the following feature/requirements:

1. A web and mobile app with a tablet view having the following features.
   - A gallery view of the record by the MRN (Medical Record Number) and the Referring MD (Medical Doctor)
   - A record search by the MRN or Referring MD
   - Record edit.
   - New record add.
   - View a report from the data collected.
   - Send message to the referring MD.
2. A Chatbot to add new records and view existing handover records.

This is a demonstration project undertaken by one individual and it is not intended to reflect any of the complexities that might arise when a larger project would be undertaken by a team of developers using a Low-Code platform.

The application requirements used for this development were shared with an in-house professional developer who used the traditional approach with native programming languages to determine how the approach taken would fulfill the criteria for comparison.

3.3. Low-Code vs Traditional Development Approach

Let us start by looking at the project execution strategy of developing an application starting from the design stage (after all the preliminary stages of planning and requirements gathering). This helps address the time management criteria of our comparison.

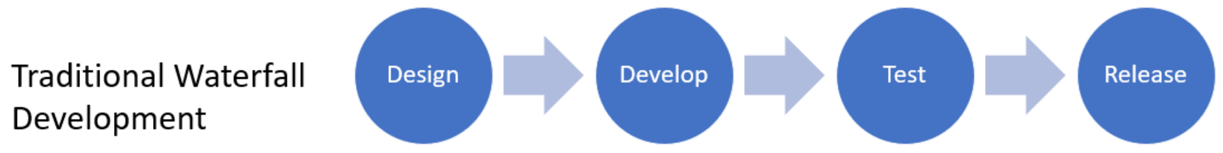


Figure 13: Traditional Waterfall Development Approach from the Design Phase

In the Traditional Dev approach (illustrated in figure 13), the process flows one way from the design phase through release with a long lead time before the release of first minimum viable product (MVP) in which the app is not deployed to most users until all development is complete. This leads to an increased risk of gaps between user initial requirements and final app release. Testing ensures the app works as designed within the agreed scope of initial agreement. However, in a traditional waterfall project, no new design happens after development to avoid ‘Scope Creep’.

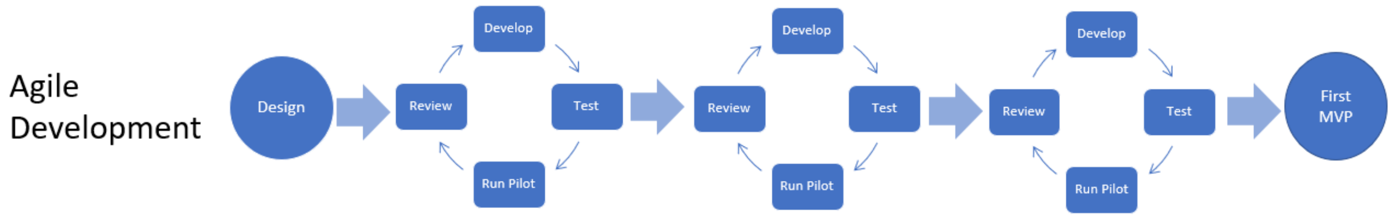


Figure 14: Agile Development Approach from the Design Phase to First MVP

In Agile development (illustrated in figure 14), there are several development sprints to develop different features of the application. Sprints are short periods, usually about two weeks long, where a scrum team works on the project to complete a set of milestones. Commonly used by software development teams, they are known to be a faster alternative to traditional waterfall development since they allow the team to get instant feedback on their work, resulting in fewer issues at the release of the product. However, this method takes a significant amount of time before the first MVP is released to users, even in a modern development approach using Agile Development with several sprints.

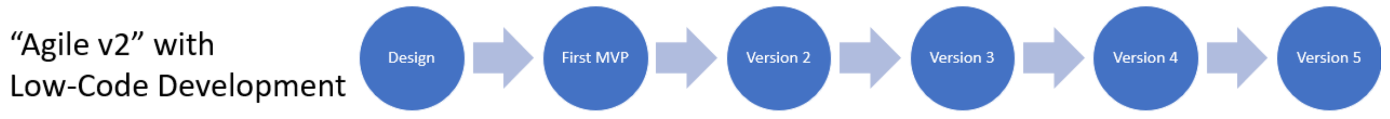


Figure 15: Low-Code Supported Agile Development Approach from the Design Phase.

Applying a Low-Code development platform in agile development (illustrated in figure 15), allows for faster release of the first Minimum Viable Product (MVP). Low-Code allows for a What-You-See-Is-What-You-Get (WYSIWYG) rapid development experience that supports the rapid release of new versions and updates to the product as new requests/requirements are received. Note that any new requests in Waterfall and Agile development typically delay development significantly. As a consequence, they are often not permitted until after the first release of the application.

3.3.1. Technical Architecture

The traditional development approach and Low-Code approach have similar architectural landscapes. In the traditional approach (as illustrated in figure 16), front-end development involves code executed on the client side and the development of the user interface which plays a major part in the user experience, while backend development is for data storage and business logic.

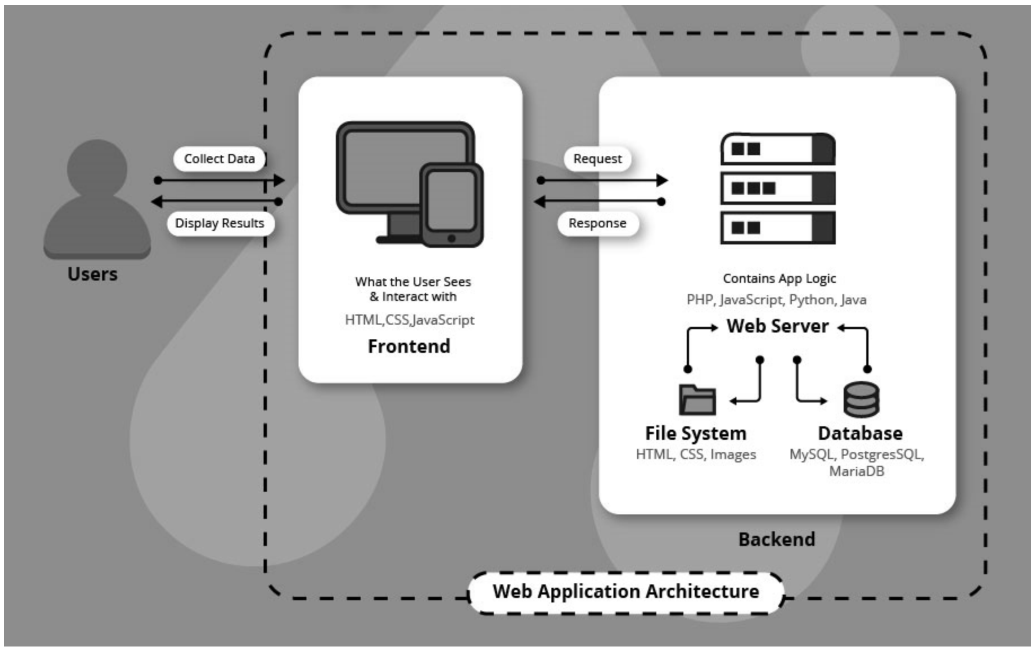


Figure 16: Typical Web Application Architecture. (Source: Singh, C. (2020, July 14). Fundamentals of Web Application... Appventurez. https://www.appventurez.com/blog/web-application-architecture)

The execution of the web application architecture requires the knowledge of several programming languages simultaneously. It is therefore essential that a good professional developer be proficient in a number of programming languages like HTML, CSS, JavaScript for front-end programming, along with PHP, Python, Java, C# for back-end programming, and SQL scripting for database manipulation.

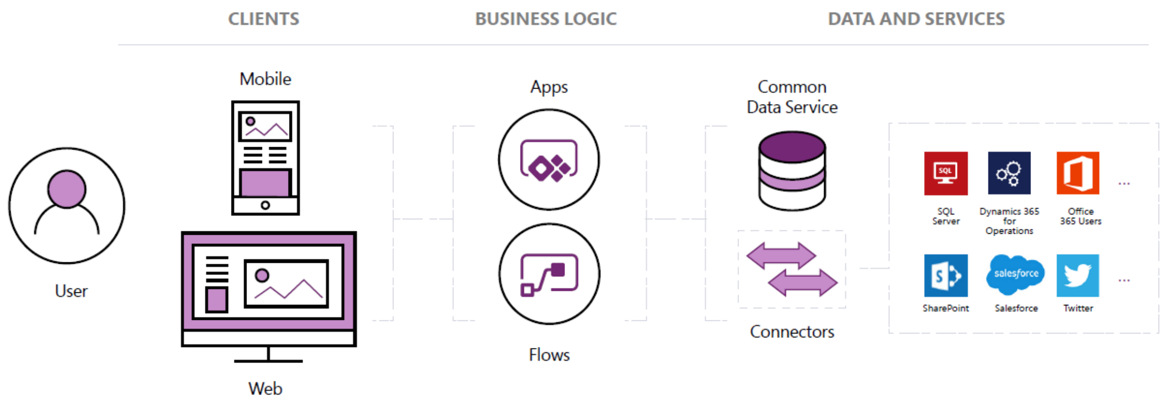


Figure 17: Architectural Structure of a Low-Code Development Platform (Source: Microsoft Inc.)

A similar architecture applies in Low-Code development (as illustrated in figure 17). Its architecture is composed of the following sections: the client, business logic, and the data and services. The low-code platform provides a graphical interface for app makers and developers to create client-facing user interfaces in both mobile and web versions.

In low-code development platforms like the Microsoft Power Platform, business logic is written in Excel-like formulas and expressions (Power FX), unlike traditional development approaches where logic is created using programming languages like C#, Java, PHP, etc. Additionally, the low-code app logic can be extended by calling out code built with programming languages or prebuilt connectors provided by the vendor.

Traditional application development depends on some sort of Database Management System (DBMS) and service such as SQL Server, Oracle DB, etc. On the other hand, a Low-Code development platform like the Microsoft LCDP offers the option of using the built-in data management system known as Dataverse (formerly Common Data Service) built on the trusted cloud platform Microsoft Azure or integrated with the organization's main database management system. This is possible since the platform has a large pool of over 240 connectors that can be integrated easily with data sources and services such as SQL server, Oracle DB, Salesforce, SharePoint etc.

3.3.2. PCU App Development

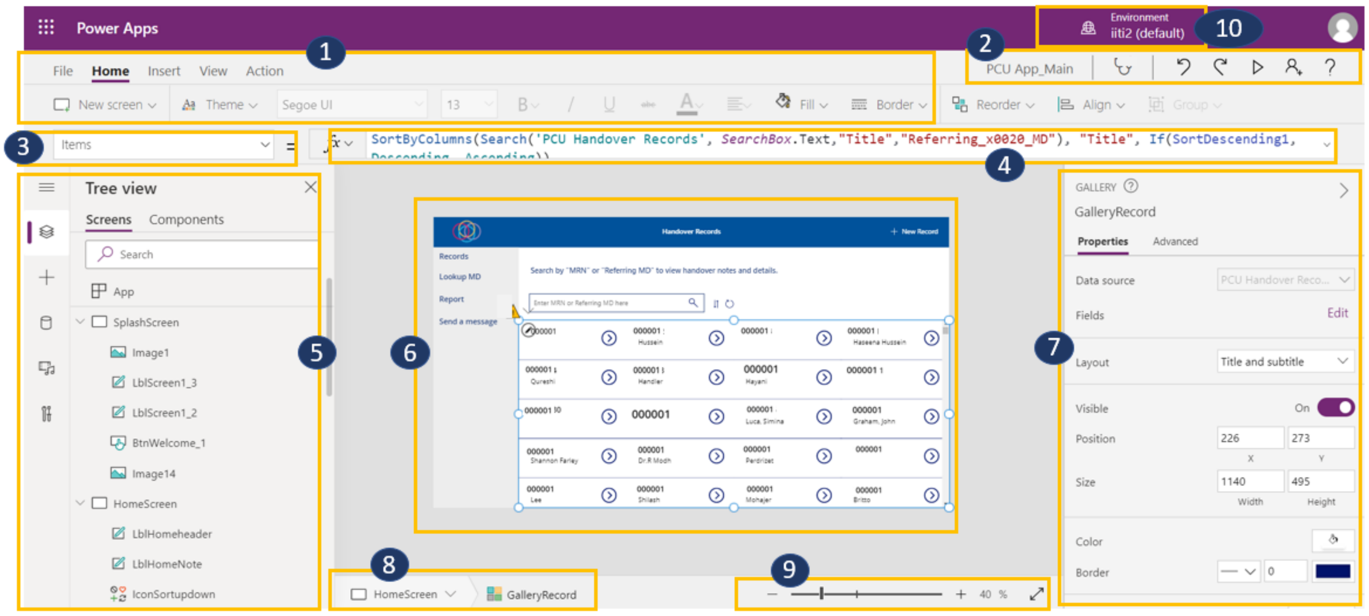
The first phase of the Palliative Care unit handover application project was the transition from the previous Excel data collection tool to a cloud-based data repository on the Microsoft SharePoint platform. This is a solution under the Microsoft 365 suite of products that provides a platform for proper document management, collaboration, security, and permission. This was followed by application development using the Microsoft LCDP with an existing data source (SharePoint) within the WOHS cloud infrastructure. The conceptualization of the app took approximately two - five days, and development using the Microsoft LCDP took approximately five days which was below the initially estimated time frame of 10 days. Documentation duration also took less time than initially estimated. See the breakdown in Table 2. Details about the development process follow.

Table 2: Estimated PCU App Development Schedule (Using the LCDP approach)

Task	**Initial Estimated Duration (Days)**	**Actual Duration (Days)**
PCU app strategy & requirement	5	5
Development	10	5
Testing	5	5
Documentation	5	3
Total	**25**	**18**

a. Build Interface:

For the PCU project, I developed a canvas app on the Power Platform. Figure 18 is the build interface of the PowerApps Low-Code platform. There are 9 main sections in the maker interface, including:



```
SortByColumns(Search('PCU Handover Records', SearchBox.Text,"Title","Referring_x0020_MD"), "Title", If(SortDescending1,
```

Figure 18: PowerApp Maker Interface

1. Ribbon – Tab and Commands
2. Commands
3. Property Selector
4. Formula Bar
5. Tree view – Screens and Components
6. App Canvas
7. Properties and Advanced Setting
8. Hierarchy Navigation
9. Zoom Dialog Box
10. Environment

1. **Ribbon – Tab and Commands**: PowerApps is very similar to the Microsoft Office interface. The ribbon has 5 tabs;

- **File** tab leads to the system menu where the developer can save, share, or publish the app.
- **Home** tab is the area where the developer can add new screens, change color themes, format and align objects.
- **Insert** tab is used to add components and objects like buttons, galleries, labels, forms, images, icons etc. to the app. It also allows adding advanced AI Builder components (such as receipt processor, form processor, business card reader and text recognizer) and Mixed Reality.
- **View** tab allows viewing, adding and refreshing Data Sources to the app to build the logic. It also allows viewing collections, media and other variables used in building the app as well as giving access to advanced setting section for app properties.
- **Action** tab allows application of logic and adding actions from Microsoft flow to selected object properties.

2. **Commands:** This tab holds the app check that checks for errors as the developer builds, undoes, redoes, previews and shares commands.
3. **Property Selector and Function Bar:** Both tabs work together to create much of the functionality and logic behind the app. Each object used in the app has a set of properties that are assigned values or controls in the function bar, using Power FX functions and commands. An instance is the button object used in the welcome page. To add a command or value to make it navigate to the Records page of the app, I selected the

button and on the property selector, picked the OnSelect property, and on the function bar entered the function:

```
OnSelect = Navigate(HomeScreen);Transition.Pop
```

4. **Tree view – Screens and Components**: Displays a tree view of all the screens in the app along with the objects they contain.
5. **App Canvas**: This is where the design of the various screens of the app is done showing the selected screen and objects within it.
6. **Property and advanced setting**: This section is like a user-friendly view of the Property selector and function bar. Properties of the selected object on the app can be customized here.
7. **Hierarchy Navigation:** Shows the structure of objects that are nested with other objects in them.
8. **Zoom Dialog Box:** This allows zooming into the app canvas in order to easily adjust or edit objects in the app.
9. **Environment:** A container that stores and manages apps, data and flows (Business logic) with roles, security, policies and target audience.

Screenshots of the final PCU app appear below in Figures 19 to 29:

Welcome Page:

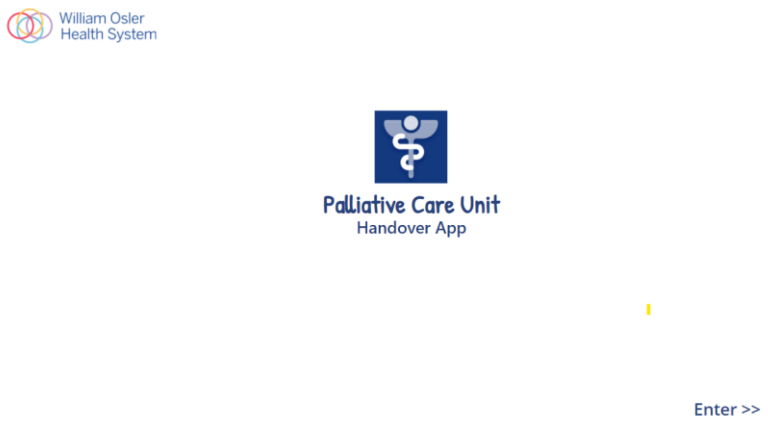
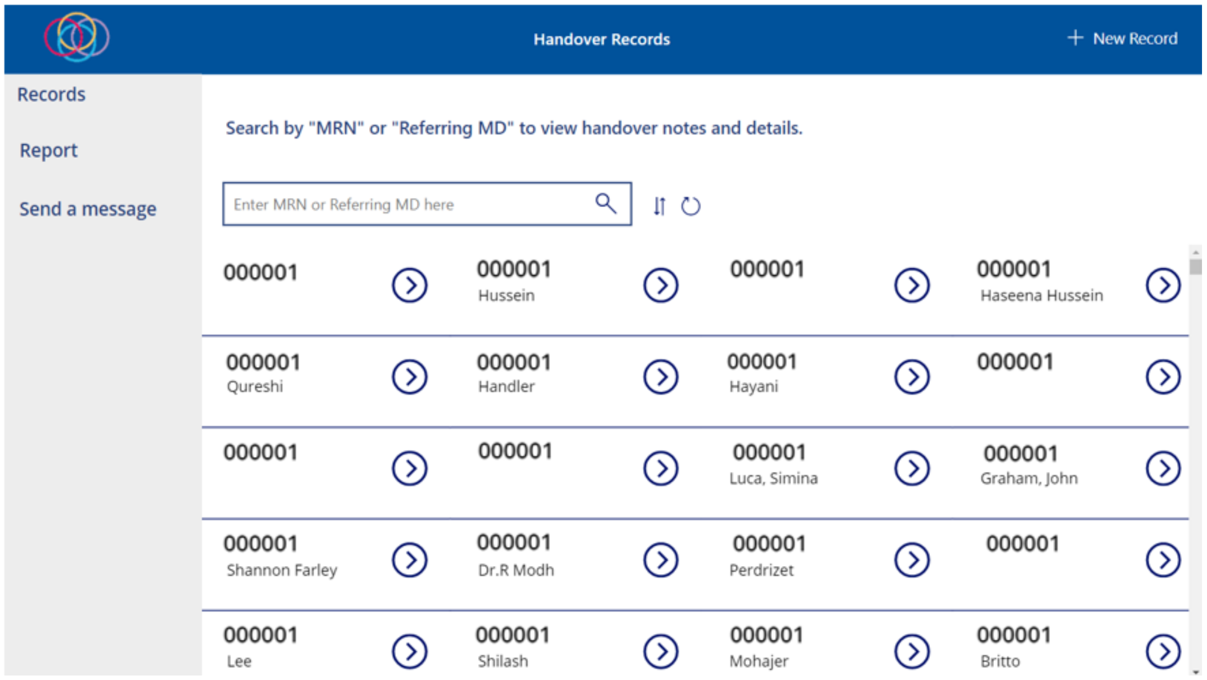


Figure 19: PCU App - Welcome Page

Records page: Provides a view of the Handover records from the data source showing the patient's MRN and Referring MD. Note that the MRN has been changed in Figure 20 - 28 for data protection.

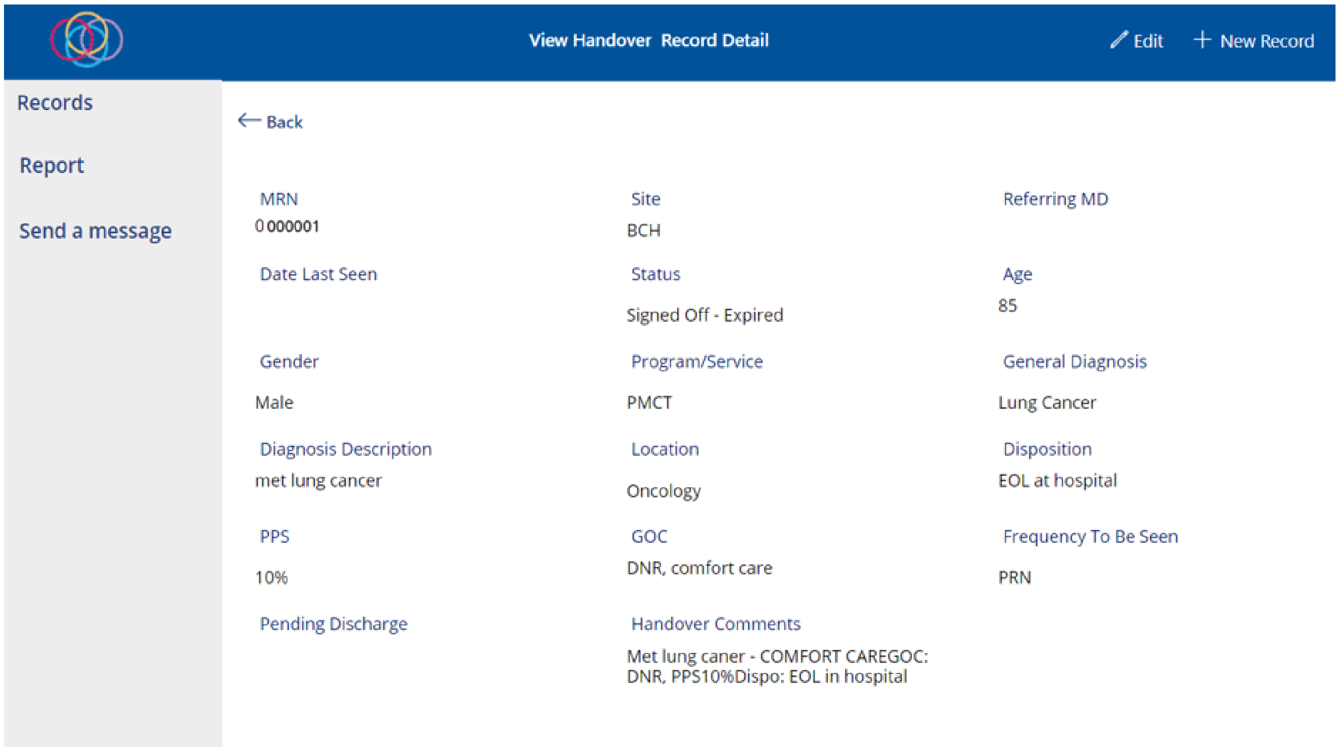


Search by "MRN" or "Referring MD" to view handover notes and details.

000001	000001 Hussein	000001	000001 Haseena Hussein
000001 Qureshi	000001 Handler	000001 Hayani	000001
000001	000001	000001 Luca, Simina	000001 Graham, John
000001 Shannon Farley	000001 Dr.R Modh	000001 Perdrizet	000001
000001 Lee	000001 Shilash	000001 Mohajer	000001 Britto

Figure 20: PCU App - Handover Records Page

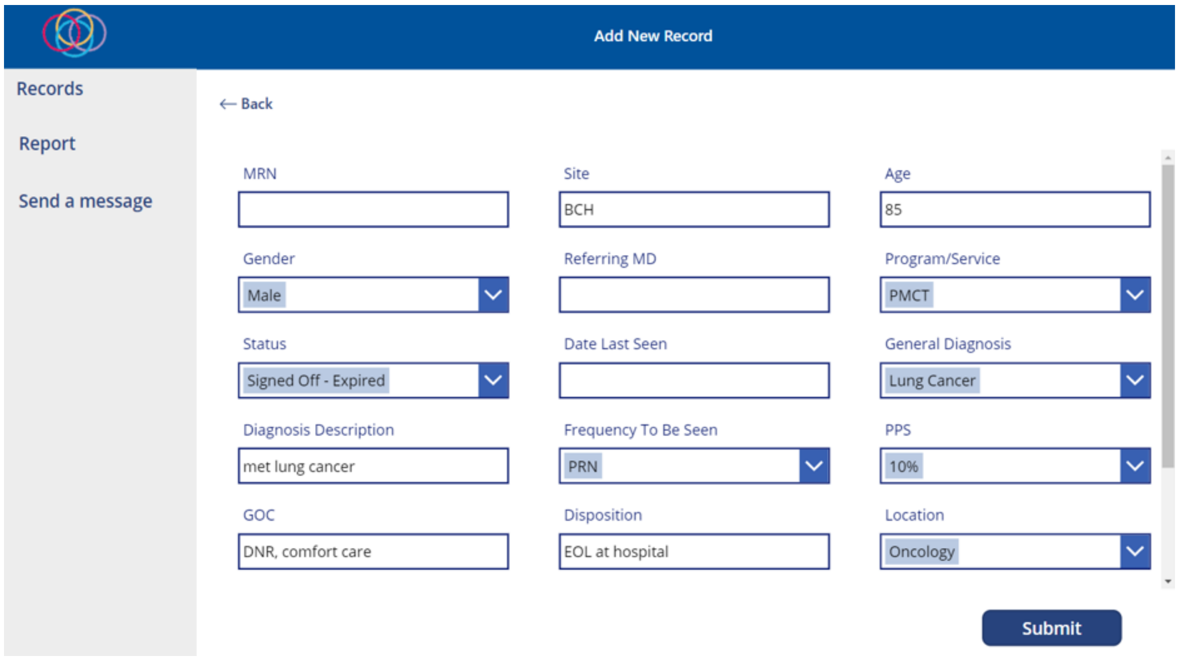
View Handover Record detail: This page shows details of the handover notes which can be edited and updated only by staff with appropriate edit permissions inherited from data source.



MRN	Site	Referring MD
0 000001	BCH	
Date Last Seen	Status	Age
	Signed Off - Expired	85
Gender	Program/Service	General Diagnosis
Male	PMCT	Lung Cancer
Diagnosis Description	Location	Disposition
met lung cancer	Oncology	EOL at hospital
PPS	GOC	Frequency To Be Seen
10%	DNR, comfort care	PRN
Pending Discharge	Handover Comments	
	Met lung caner - COMFORT CAREGOC: DNR, PPS10%Dispo: EOL in hospital	

Figure 21: PCU App - View Handover Record Page

Add new record: page allows staff to create a new handover record.



MRN	Site	Age
	BCH	85
Gender	Referring MD	Program/Service
Male		PMCT
Status	Date Last Seen	General Diagnosis
Signed Off - Expired		Lung Cancer
Diagnosis Description	Frequency To Be Seen	PPS
met lung cancer	PRN	10%
GOC	Disposition	Location
DNR, comfort care	EOL at hospital	Oncology

Figure 22: PCU App – Add New Record Page

Report (see Figure 23): this page displays a report of the general diagnosis by frequency at the Palliative Care Unit, providing insights into the diagnosis with the highest frequency at the unit.

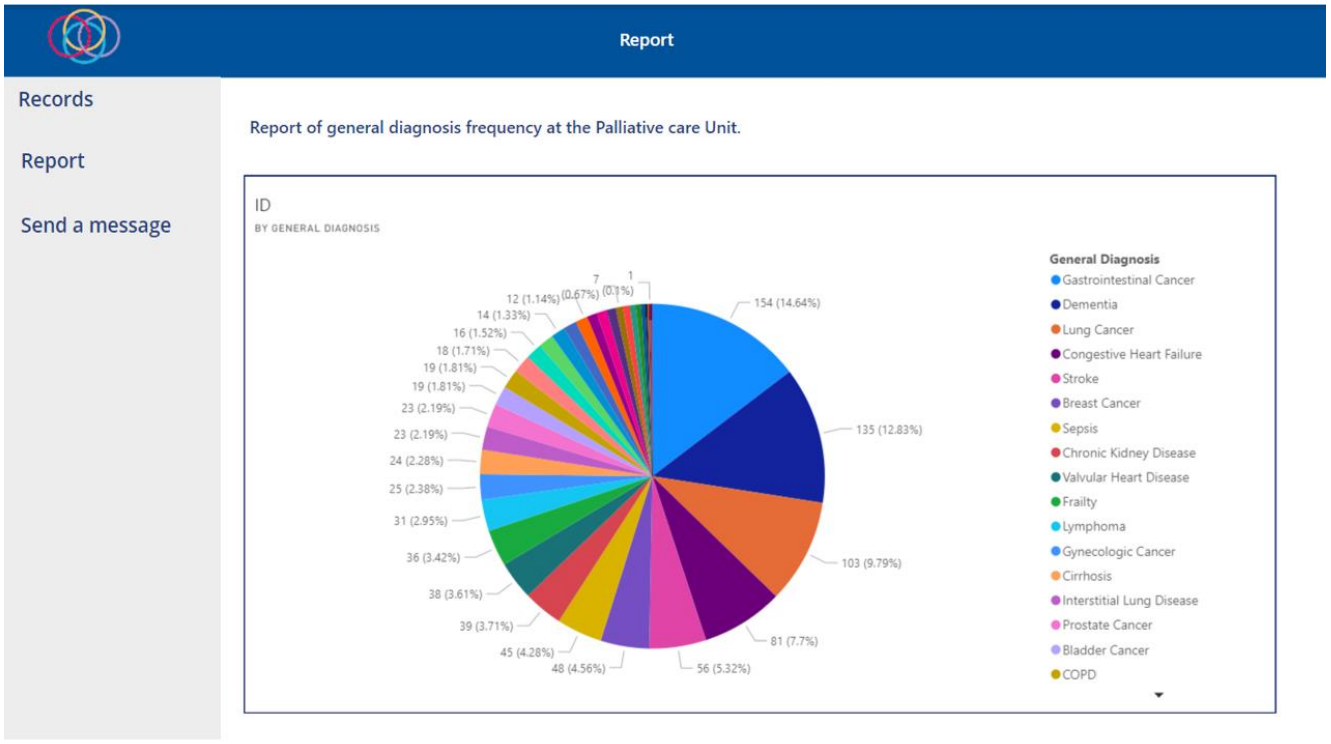


Figure 23: PCU App- Report Page

Send Message: This page allows a staff member to instantly send a message to an MD or fellow staff member from the app. This allows MDs to quickly communicate questions or thoughts to a referring MD.

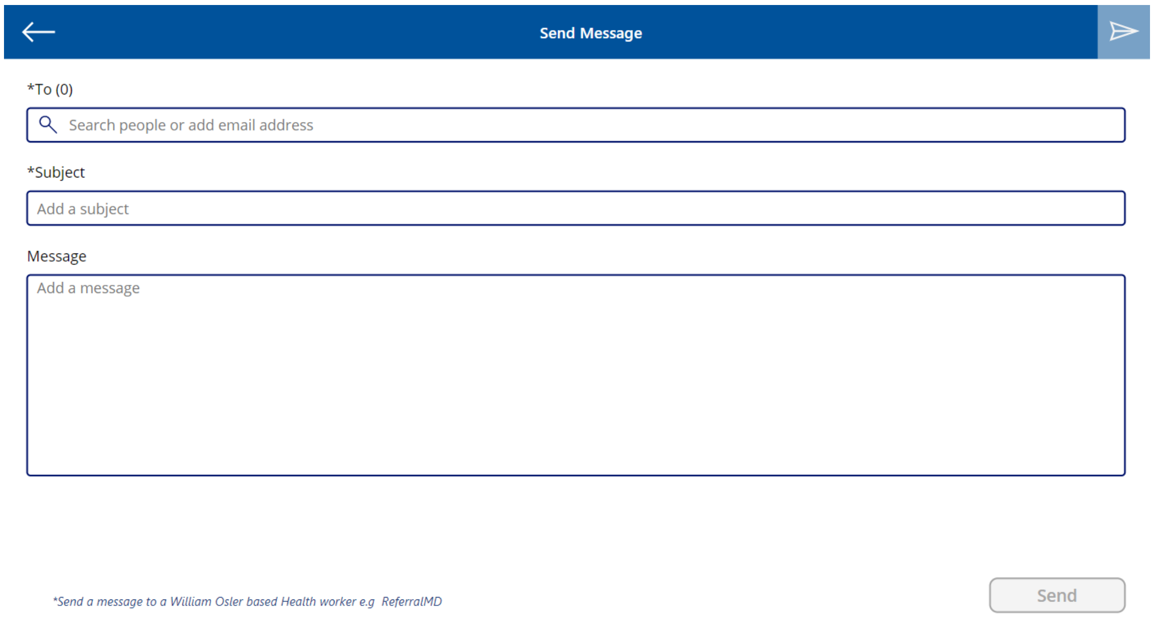

Send Message
*To (0)
Search people or add email address
*Subject
Add a subject
Message
Add a message
*Send a message to a William Osler based Health worker e.g ReferralMD
Send

Figure 24: PCU App – Send Message Page

App Publishing:

The PCU app can be published and consumed by multiple means:

- By the ‘Power App’ mobile application on Android or iOS platforms.
- By the Microsoft Teams platform
- By Microsoft SharePoint

The published app can be accessed through the ‘Power App’ application for mobile (Android and iOS platforms). Users would have to download the Power App on their mobile device to use the application created on Power Apps. Once downloaded, the user will be required to go through multi-factor authentication on the Microsoft network to be logged in. Once authenticated the user can see a list of applications built on the platform that they have access to.

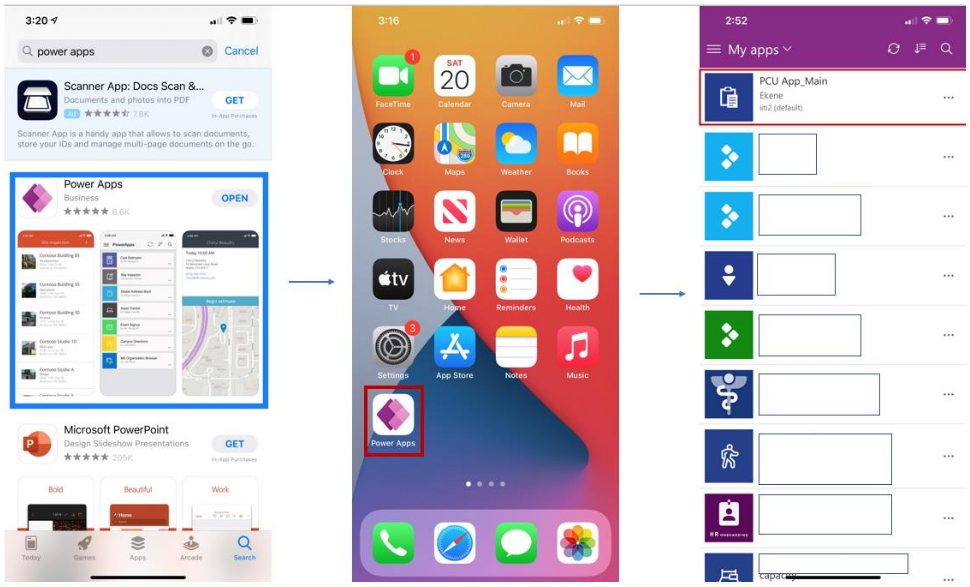


Figure 25: Power Apps Mobile Apps Installation Screens

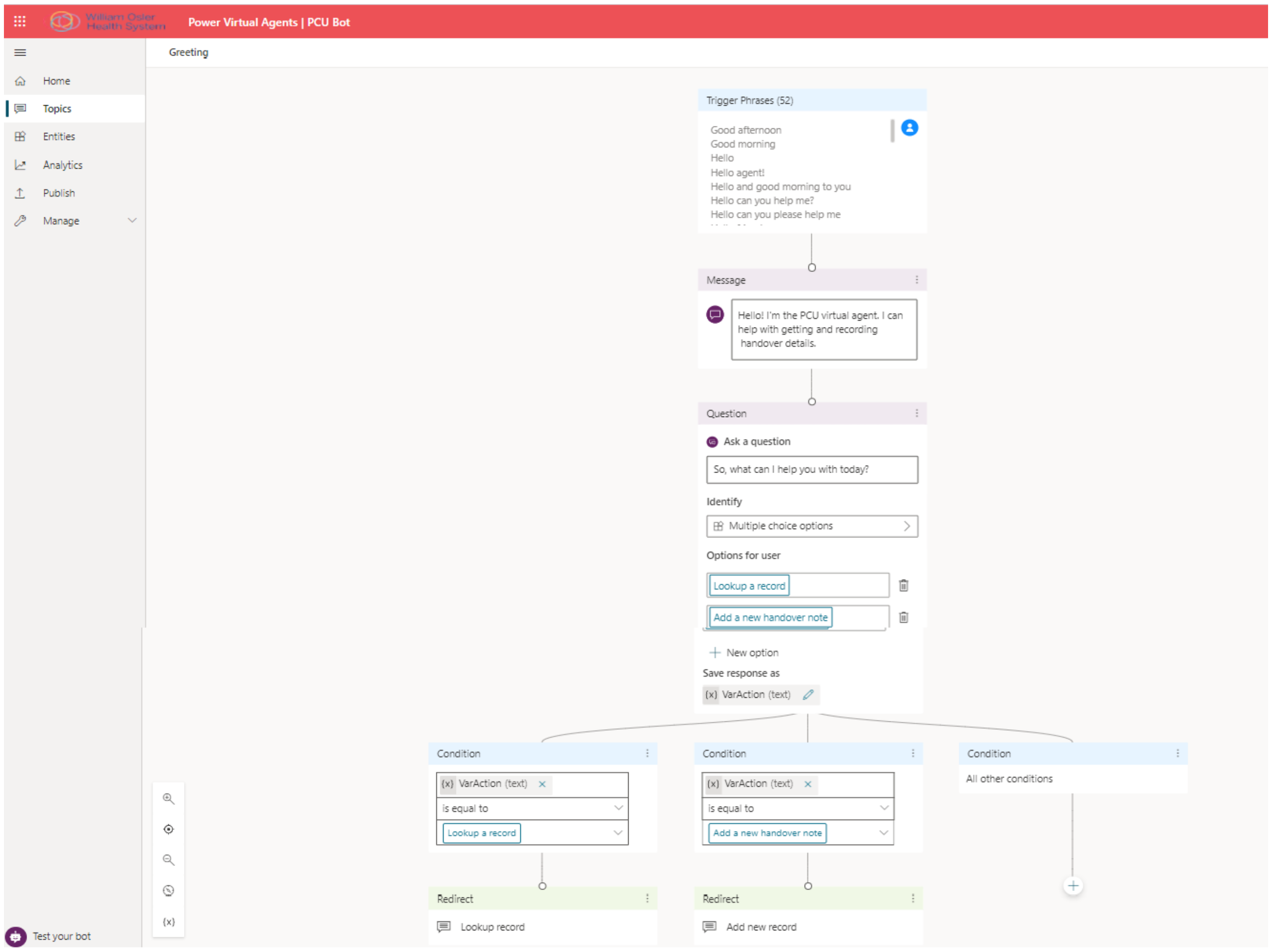
To use the apps on MS Teams and on SharePoint, the organization’s system administrator must configure and add the app to Teams and SharePoint to be consumed by end users.

Once launched, users get the welcome screen where they click on the ‘Enter’ button. Once in they are taken to the Records page.

Chatbot Development:

The Chatbot feature of the PCU app includes the ability to add a new record and view a record using the Medical Record Number (MRN). The Chatbot was built with a combination of the Microsoft LCDP tools which include Power Virtual Agent for the bot development, Power Automate to integrate the flow between the variables in the virtual agent and the data list in SharePoint, and Microsoft Team where users can trigger and interact with the bot.

Figure 26 Shows the Power Virtual Agent interface where the chatbot is built and authored step by step.



Trigger Phrases (52)

Good afternoon
Good morning
Hello
Hello agent!
Hello and good morning to you
Hello can you help me?
Hello can you please help me

Message

Hello! I'm the PCU virtual agent. I can help with getting and recording handover details.

Question

Ask a question

So, what can I help you with today?

Identify

Multiple choice options

Options for user

Lookup a record

Add a new handover note

New option

Save response as

(x) VarAction (text)

Condition

(x) VarAction (text)
is equal to
Lookup a record

Redirect

Lookup record

Condition

(x) VarAction (text)
is equal to
Add a new handover note

Redirect

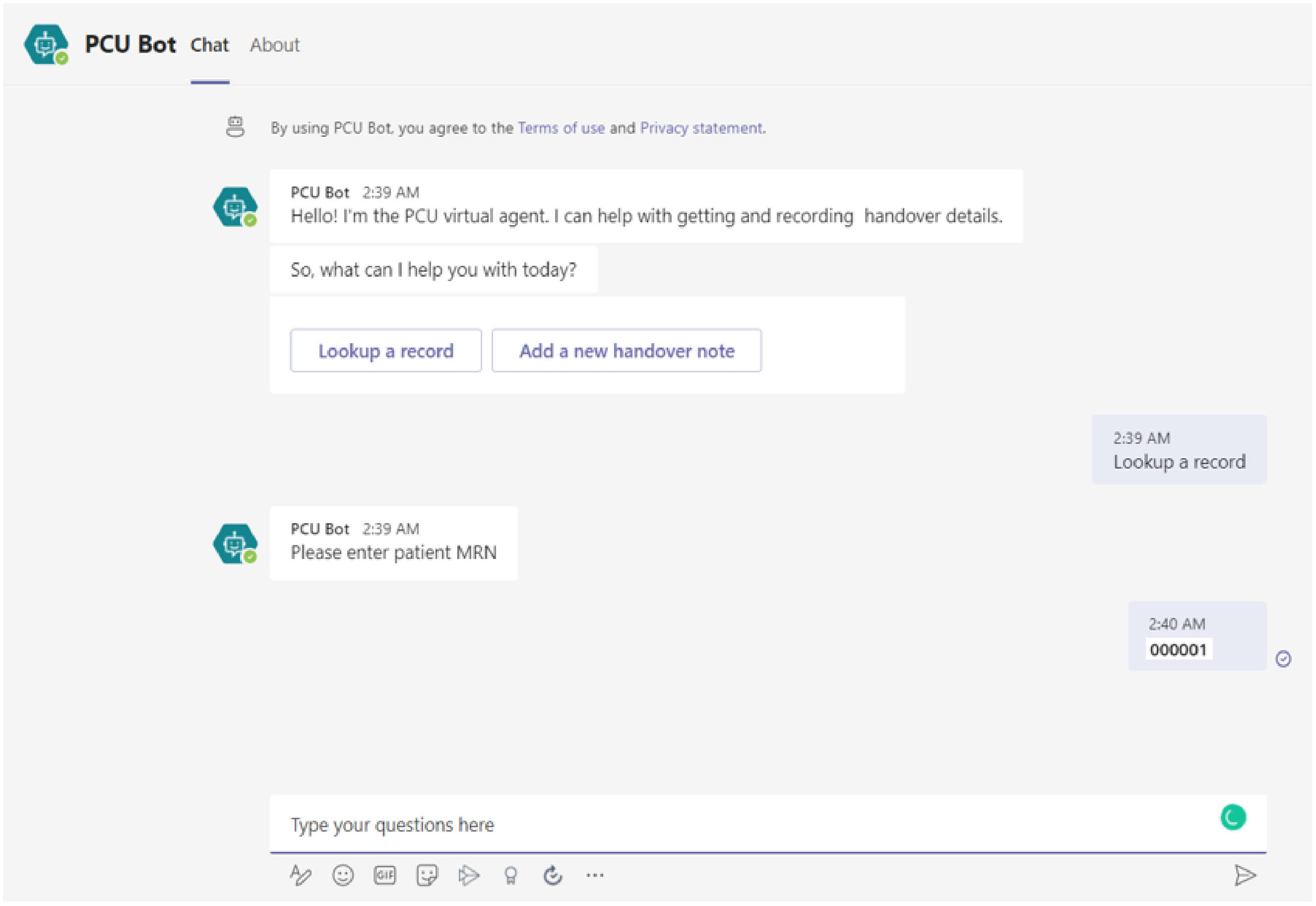
Add new record

Condition

All other conditions

Figure 26: Power Virtual Agent (PVA) bot authoring interface

Figure 27 shows the Chatbot interface when integrated with Microsoft Teams. This allows for easy accessibility and interaction with the bot.



By using PCU Bot, you agree to the Terms of use and Privacy statement.

PCU Bot 2:39 AM
Hello! I'm the PCU virtual agent. I can help with getting and recording handover details.

So, what can I help you with today?

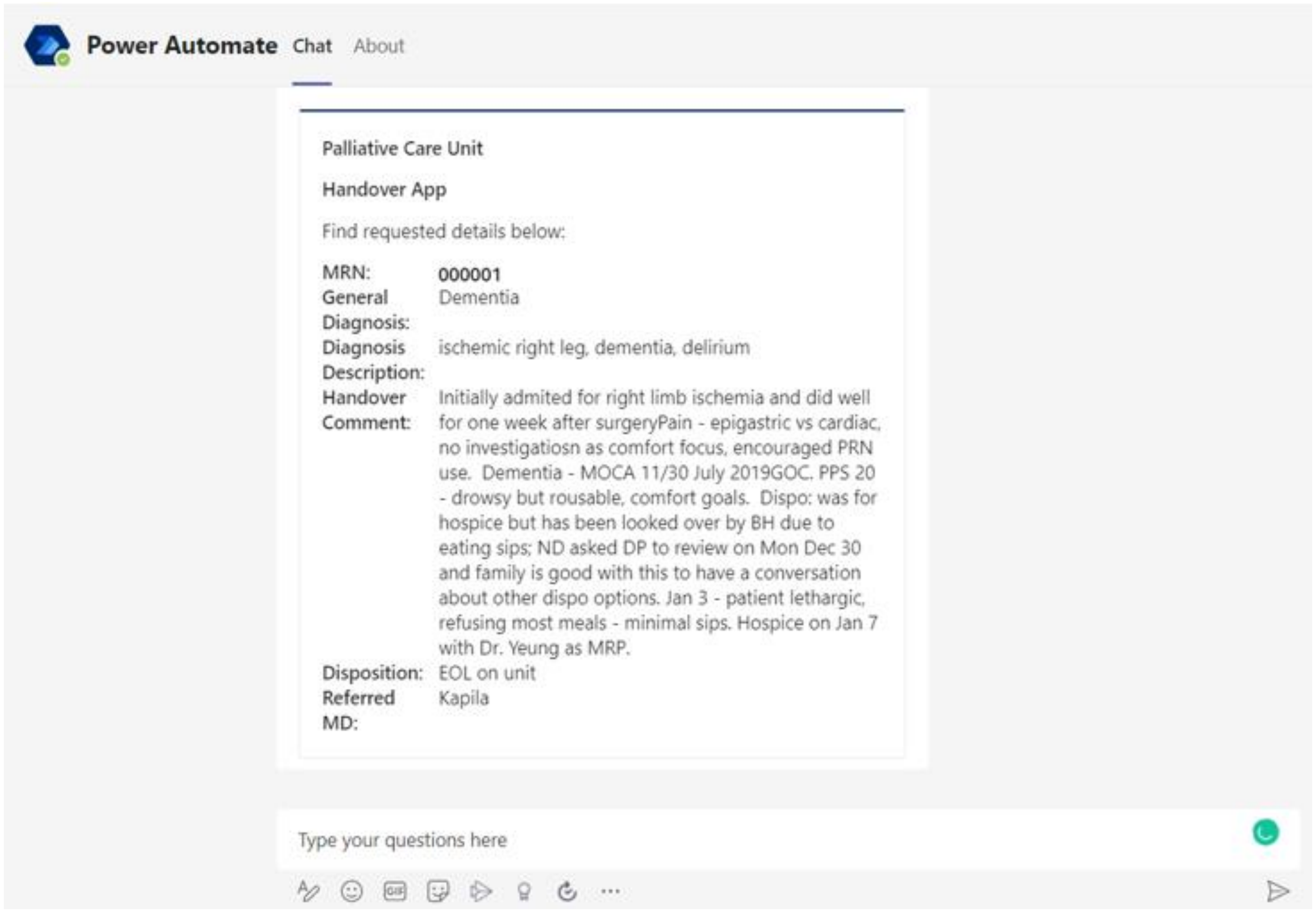
2:39 AM
Lookup a record

PCU Bot 2:39 AM
Please enter patient MRN

2:40 AM
000001

Figure 27: Chatbot integrated in Microsoft Teams.

Figure 28 Shows the result of the PCU handover record when the patient MRN is entered. The display format was created with Adaptivecard.io designer, a Microsoft tool that is used to design snippets of UI and integrate in JSON format into major platforms. Figure 29 shows the Adaptive Cards interface.



Palliative Care Unit

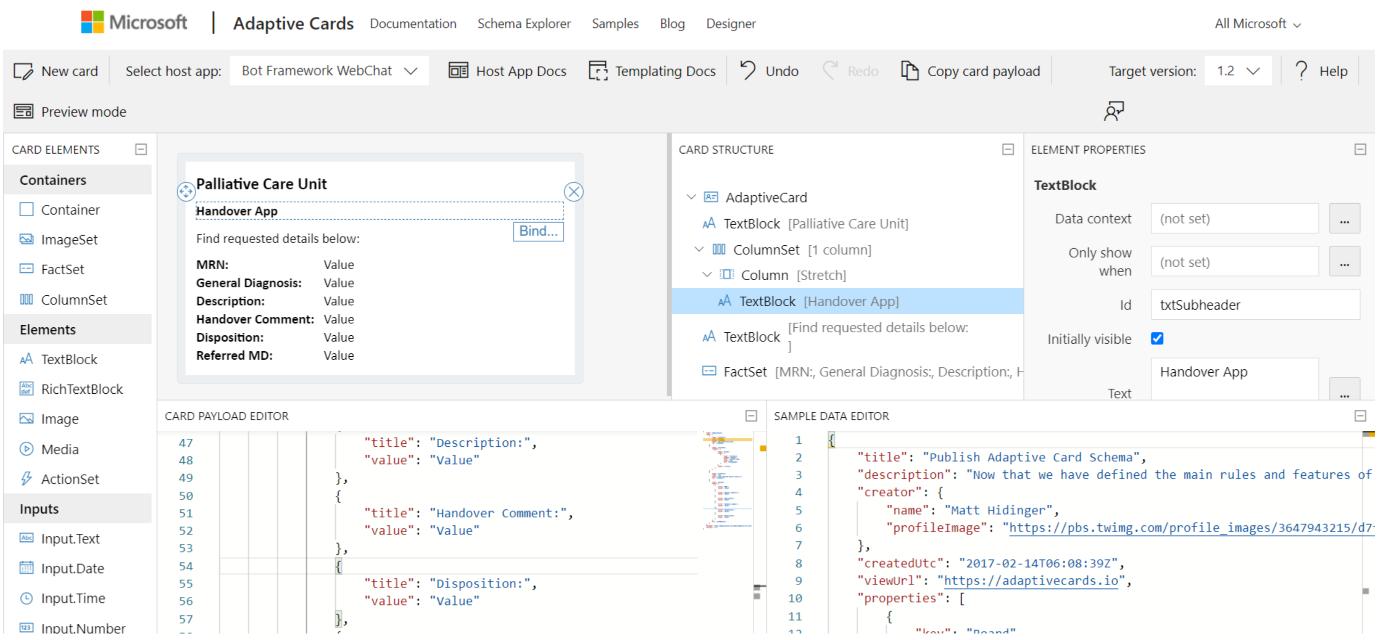
Handover App

Find requested details below:

MRN:	000001
General Diagnosis:	Dementia
Diagnosis Description:	ischemic right leg, dementia, delirium
Handover Comment:	Initially admited for right limb ischemia and did well for one week after surgeryPain - epigastric vs cardiac, no investigatiosn as comfort focus, encouraged PRN use. Dementia - MOCA 11/30 July 2019GOC. PPS 20 - drowsy but rousable, comfort goals. Dispo: was for hospice but has been looked over by BH due to eating sips; ND asked DP to review on Mon Dec 30 and family is good with this to have a conversation about other dispo options. Jan 3 - patient lethargic, refusing most meals - minimal sips. Hospice on Jan 7 with Dr. Yeung as MRP.
Disposition:	EOL on unit
Referred MD:	Kapila

Type your questions here

Figure 28: Displayed Results of the Handover Comment.



Palliative Care Unit

Handover App

Find requested details below:

MRN:	Value
General Diagnosis:	Value
Description:	Value
Handover Comment:	Value
Disposition:	Value
Referred MD:	Value

TextBlock

Data context	(not set)
Only show when	(not set)
Id	txtSubheader
Initially visible	✓
Text	Handover App

```
                "title": "Description:",
                "value": "Value"
            },
            {
                "title": "Handover Comment:",
                "value": "Value"
            },
            {
                "title": "Disposition:",
                "value": "Value"
            },
```

```
{
    "title": "Publish Adaptive Card Schema",
    "description": "Now that we have defined the main rules and features of
    "creator": {
        "name": "Matt Hidinger",
        "profileImage": "https://pbs.twimg.com/profile_images/3647943215/d7
    },
    "createdUtc": "2017-02-14T06:08:39Z",
    "viewUrl": "https://adaptivecards.io",
    "properties": [
        {
```

Figure 29: Adaptive Cards Interface (adaptivecards.io)

CHAPTER 4: FINDINGS

Based on the comparison carried out between Low-Code development and the traditional development approach, following the key criteria areas mentioned in the previous chapters, the following findings address the two supporting research questions:

RQ2: Why has Low-Code not been considered an appropriate approach for enterprise/healthcare application development? OR What are the concerns around using LCDP to develop applications for the healthcare system?

RQ3: What are the benefits of using Low-Code Development Platforms to develop applications for healthcare system management?

The results of the comparison are based on the practical development of the Palliative Care Unit app done on a Low-Code development platform and an interview with a developer who uses the traditional app development approach to create applications.

4.0.1. General

Time is a critical factor in implementation or integration of product. If more time is spent on analyzing and planning the business process, then implementation is much smoother and will require less time troubleshooting. Time should also be spent concentrating on making the product mature and flexible, so it is easy to adapt for future requirements. More time spent on business analysis provides the opportunity to understand all the dependencies, risks and any bottlenecks that may occur during the development process.

The Low-Code approach is quicker and easier to configure and carry out integrations with a clearly defined scope. Additionally, Low-Code allows for extensions to accommodate additional user requirements without disrupting the entire development plan.

The Traditional/Custom approach requires all of the above steps plus development and testing time. Making a product mature so it can cater to user needs and serve its main purpose requires time and dedication from all stakeholders. If a more modular approach is adopted, the traditional approach becomes more desirable as it becomes more flexible to adapt to future needs. The traditional approach has more implementation steps than the Low-Code approach, but some steps can run in parallel such as testing and development which can run hand in hand if the Agile-Scrum approach is incorporated. Taking into consideration stakeholders, feedback can also be incorporated into development time. In combination with DevOps Continuous Integration/Continuous Deployment (CI/CD), this makes it a lot easier to implement changes without any real downtime.

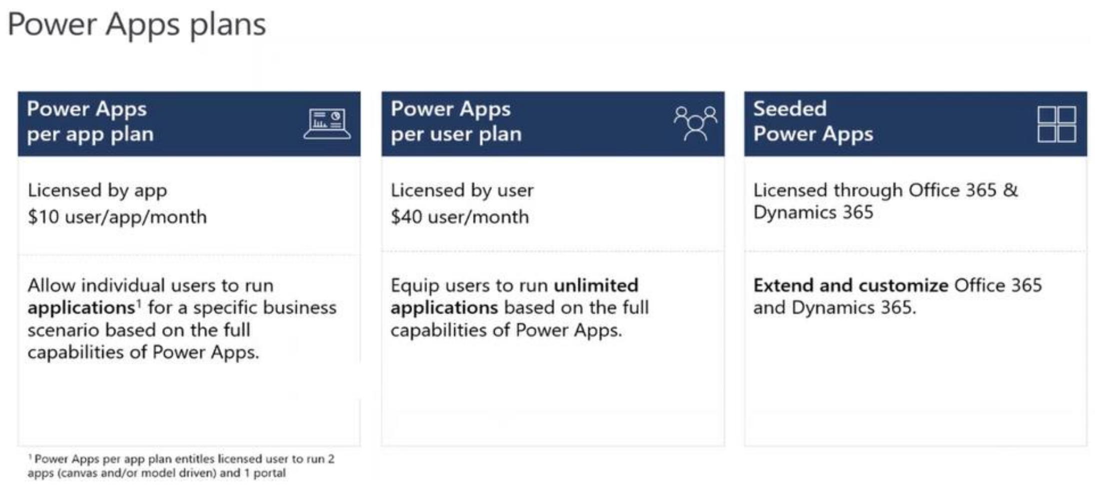
4.1. Areas of concern (Drawbacks of LCDP)

Findings showing the major areas of concern about the Low-Code approach being appropriate for the WOHS and other health systems are discussed below:

4.1.1. Cost Management

Since Low-Code Development Platforms (LCDPs) are categorized as cloud-based software development platforms delivered through a Platform-as-a-Service (PaaS) model, and cloud computing can also be categorized as an Infrastructure-as-a-Service model, there is a concern about recurring subscription costs and vendor-lock in. For this thesis, the Low-Code and Cloud platform used was the Microsoft 365 platform which has different subscription plans (see figure 30) that organizations can sign up for. This subscription model is either a Per User plan, Per

App plan or a Seeded plan that comes with Office 365 E3 or E5 license approaches. Several conversations with account officers at Microsoft determined what it would cost to adopt their LCDP as a development tool for WOHS. A large organization like WOHS with over 5000 staff would require a subscription cost of $40 for every user who used the apps built with the platform. This is a major area of concern and a drawback. We also looked up other LCDPs and found that they had similar licensing models. This confirms that adopting a LCDP would require a considerable amount of annual financial investment.


Power Apps plans

Power Apps per app plan	Power Apps per user plan	Seeded Power Apps
Licensed by app $10 user/app/month	Licensed by user $40 user/month	Licensed through Office 365 & Dynamics 365
Allow individual users to run **applications**[1] for a specific business scenario based on the full capabilities of Power Apps.	Equip users to run **unlimited applications** based on the full capabilities of Power Apps.	**Extend and customize** Office 365 and Dynamics 365.

[1] Power Apps per app plan entitles licensed user to run 2 apps (canvas and/or model driven) and 1 portal

Figure 30: Microsoft Power Plans (Source: Microsoft Inc. 2021)

When we compared the LCDP costs with the traditional approach, we realized that the traditional development approach involves engaging the services of in-house developers/coders with its own set of costs. However, if open-source products could be used then cost could be less due to savings from licencing fee avoidance. This approach involves a thorough cost analysis of development costs, testing costs, hosting/hardware costs and maintenance costs. If more time is spent on analysis there is less development time and hence lower development cost. Hosting/ hardware cost could be controlled with self-hosting infrastructure or cloud computing. A more incremental approach or on-demand approach could be adopted to keep the costs in check, and if there is no usage licence the only cost that is going to increase arises from scaling the system up. However, additional costs can be incurred in the traditional approach if DevOps tools for testing, monitoring and deployment are purchased. Overall, we found that the traditional approach offers more options such as using either open-source or paid tool alternatives during the application development lifecycle. However, that also means there would be a collection of third-party applications that would need to be managed by the IT team.

4.1.2. Culture Change and Training

While adopting the LCDP to develop the PCU Handover App, it was necessary for staff in the IITI unit at WOHS to be designated for special training on the new platform. It was important for developers using a Low-Code development approach to get familiar with using the platform. I joined the IITI team with some prior knowledge of Microsoft products. However, upon joining the team, I had to take some training on the Microsoft LCDP 'Power Platform' which was straight-forward for me to adopt to as a developer. This training took a few weeks in addition to training sessions with the Microsoft product team. This approach illustrates that it can be less disruptive to have a few "champions" in the IITI unit who can develop smaller apps until there is enough experience among the developers to move on to larger apps. This strategy was very helpful in getting the champions accustomed to creating apps using the LCDP to write to an on-premises database as well as to read data and information from the database. After creating apps for different test scenarios, the developers, (in this case champions) were able to create the PCU app in the production environment. Management support for the use of the new approach to application development was also a critical area of concern, as PCU apps were already being used to replace the existing traditional approach to application development.

4.1.3. Vendor Lock-in

Another area of concern was the potential of being locked-in to a particular LCDP provider. This comes with its benefits and drawbacks. From our findings, we realized that an advantage of working with an LCDP product provider comes with easy accessibility to the vendor's support engineers who are on stand-by and can provide support with training, resolving technical issues, and maintenance of the infrastructure. However, there is the unpredictable nature of working with a vendor, where subscription plans and prices can be changed at the vendor's discretion. Additionally, it may be difficult to integrate some LCDP with other large pre-existing systems within the organization. These are important issues to be considered during vendor selection.

4.1.4 Summary

This study has identified three major areas of concern around using LCDP to develop applications for the healthcare system. Table 4 provides a summary of these concerns.

Table 4: Overview of Areas of Concern in the Adoption of LCDP

Areas of concern in the adoption of LCDP	
Cost	The cost of licensing and subscription is a major cause of concern as it is not a fixed price and can be increased at the discretion of the product provider.
Culture Change & Training	There would be a learning curve associated with adopting any LCDP
Vendor Lock-in	Concerns around vendor unpredictability and control as well as integration with other large systems already existing in the organization. These concerns are magnified due to vendor lock-in.

4.2. Benefits of Using LCDP

4.2.1. Time Management/Speed

For this thesis, speed was a critical criterion for comparison between the two development approaches to deliver applications in the health system. Low-Code has proven to be a more rapid approach to application development as the development of the Palliative Care Unit app was completed within days after product conceptualization and scoping. Low-Code replaces the rigidity of traditional development with agility and speed.

The use of formulas and expressions in the development of applications using low-code or in some cases no-code at all to build logic into an application makes the traditional development a less enticing option. One of the major drawbacks with traditional programming is that of backend code documentation and debugging which makes it difficult for a programmer to pick up another programmer's coding project. However, in the case of Low-Code, a developer can easily pickup from another developer's project as the logic is easy to read and saves time.

Table 5: Comparison of Estimated Duration for the PCU App Development for the Traditional and Low-Code Development Approach

Task	**Traditional Dev. Duration (Days)**	**Low-Code Dev. Duration (Days)**
PCU app strategy & requirement	5	5
Development	15	5
Testing	5	5
Documentation	5	3
Total	**30**	**18**

Table 5 gives an overview of the estimated comparison of time it would take to complete the PCU App between the Native traditional approach and the LCDP approach. The estimates for the LCDP duration were based on the time it took for me to develop the PCU App and the duration of the traditional approach are estimates provided by developer Gurinder Bassi, based on an equivalent case study.

4.2.2. Risk Management/Security

Regardless of which development approach is taken, security is very critical and data protection is of utmost priority. Low-Code provides a more effective way of including security right from the beginning of the development process in a security sensitive environment which supports the concept of DevSecOps. From our findings, the Microsoft LCDP integrated with already existing security measures, including identity and access management, and using the Azure Active Directory of the Microsoft Cloud platform (Azure). The platform has layers of security that protect data privacy and integrity by implementing user-based and role-based security and policy governance, and it supports multi-factor authentication within an isolated secure development environment. This, together with the hospital's existing security protocols that support the full lifecycle of protected health information (PHI) gives an extra assurance of security compliance.

The traditional approach requires identifying critical bugs and loopholes during the development and testing phases. If tried and tested approaches are adapted in SDLC this makes security and risk management more prudent and less error prone. Coding standards are also very critical and during the testing phase every single module must be tested and then securely integrated. The traditional development approach requires not only software design but also security model implementation during development. With this approach, developers can leverage some of the security products available in the market such as encryption software.

4.2.3. Human Resource Management

Usually, a full-scale development team consists of a number of team members or stakeholders, including the Project Manager, Product Manager, Product Owner, Team lead, Tech lead, Full-Stack Developer, Front-end Developer, Back-end Developer, Quality Assurance lead, Tester and UI/UX Designer. For a traditional development approach, some or all of the expert development team would need to be hired to work on the project when required. This comes with additional costs, depending on the structure around the project or the size of the project, and the contract type with the hired developers. Our findings show that a Full-Stack developer in Canada gets paid in the range of CA$71,000/year to CA$112,000/year (Glassdoor, 2021). However, a Low-Code approach offers an opportunity to increase development talent of existing staff as it is an easier application development approach to adopt readily. With Low-Code, citizen developers or people with minimal coding skills and other IT professionals can contribute to the efforts of the professional development team within a healthcare organization. However, in the traditional approach, all development and design tasks can only be carried out by professional development programmers, adding to the cost of using a traditional development approach.

4.2.4 Communication Management & Mobility

Depending on the structure around the project or the size of the project, some or all of the development team will be required to communicate, share information and collaborate during the course of the development lifecycle. Low-Code provides a platform for easier communication, sharing and collaboration on development projects. With a low-code development platform, it is more straightforward to document the development process, providing a better transition process in development projects. Different components of the application can be developed by different developers, even in different physical locations and then they can be combined to create the Master app. Additionally, since the LCDP is built on a Cloud-service platform, it leverages the power of the cloud to provide the ability for app development by people in different locations and other team member to view the changes made to the same project by other team members. However, in the traditional development approach, developers require third-party project management tools to track and collaborate on development projects. Additionally, with multiple lines of code, it is sometimes difficult for developers to properly document their development files and transition their work to other developers. This can be a bottleneck in communicating details of the project.

4.2.5. Upgradability & Version Control

A common scenario is the need for a new version of an application. With a Low-Code development approach, creating new versions of an applications can be achieved with ease as our findings show that most LCDPs have an embedded version control system that keeps track of the latest saved modifications to an app prior to and after publishing. This makes the reversal of some changes possible as the administrator can seamlessly revert to previous versions of the app when the need arises. On the other hand, with the traditional approach of building applications solely based on code, version control would have to be managed by a third-party tool such as GitHub where reversal to a previous version of the app can lead to breaking the code. This would result in additional debugging, including proper management and code documentation.

4.2.6. Scalability

For this thesis we were able to evaluate the scalability based on connection and extension to a variety of data sources. Our findings in that regard show that the LCDP is able to connect to various data sources and third-party applications that may be required to extend the capability of the application. Figure 1A in the appendix shows some applications that can be integrated into the Microsoft LCDP. When it comes to evaluating the scalability based on performance, system process demand and scaling provisioned resources like database size and data storage services, more attention would need to be paid to the cloud service utilized by the LCDP. Since most LCDPs are cloud-based, it would be easy to scale up or down the resources used in the app creation process, while features and services can be added or removed as the case may be.

4.2.7. Usability

With LCDP, the user interface and designs are responsive and easy to use on any mobile device or web interfaces, requiring very minimal coding input to achieve a clean interface design for the app. With the traditional approach, a lot of effort goes into coding and programming in CSS and JavaScript to create a desirable design for the app, requiring considerably more time to achieve.

4.2.8. Maintainability

With the traditional approach, maintenance is multifaceted as it includes the maintenance of applications created, including every other solution integrated or used in the development of the application. On the other hand, with LCDP, the maintenance is partly tied to the support service contract provided by the product provider along with the administration and management by the internal IT team. This makes the maintenance of the LCDP very manageable.

4.2.9. Summary

This study identified benefits of using LCDPs to develop applications for the healthcare system. Table 6 provides a summary of the findings.

Table 6: Overview of Benefits to the Adoption of LCDP

Benefits to the adoption of LCDP	
Speed	Create app within days with minimal coding required for just the business logic configuration,
Security	The concept of environment isolation within the platform creates security boundaries at a level below the tenant-level that wraps all users, apps, data, and connections.
Human Resources	Other staff can be trained on the LCDP thereby releasing the time for programmers to focus on more complex tasks.
Mobility	Leverages the power of the cloud to provide the ability for app development by people in different locations and other team members can view immediately changes made to the same project.
Upgradability	Eases version control management and extension of created application.
Scalability	Able to connect to various data sources and third-party applications that may be required to extend the capability of the application.
Usability	The user interface and designs of app created with LCDPs are responsive and easy to use on any device mobile or web interfaces.
Maintainability	Partly tied to the support service contract provided by the product provider along with the administration and management by the internal IT team.

4.2.10. Overview of Comparison

To summarize the findings, Table 7 represents key criteria of comparison and the values for both the Traditional and the Low-Code development approaches.

Table 7: Summarized Comparison of Traditional and Low-Code Development Approaches

Criteria	*Traditional Approach*	*Low-Code Approach*
Cost Management	Development cost, testing, hosting/hardware cost and maintenance cost are major considerations	Recurring and unpredictable Subscription cost, Vendor lock-in are major considerations.
	Cost of hiring of multiple professional developers to develop and maintain based on full-time, contract or part-time basis cost more than the developer license cost for a LCDP per user in a year.	Learning curve and training existing staff/resources on new LCDP are major factors for adoption and cause culture change.
Time Management/Speed	The complexity around writing several lines of code and third-part tools for executing different aspects of the software development lifecycle in a traditional development approach is time consuming.	The graphical interface and drag-and-drop nature enhance usability and speeds up development time. Business logic built with declarative languages also increases speed of development time.
Risk Management/Security	Security depends solely on third part tools, services and the network infrastructure of the organization for access management.	Development is done in a securely sensitive environment with multi-factor authentication and supports the concept of DevSecOps for testing, monitoring, and security maintenance.
Communication/ transition Management	With multiple lines of code, it is sometimes difficult for developers to properly document their development files and transition their work to other developers. This can be a bottleneck in communicating details of the project.	Most low-code provides a platform for easier communication, sharing and collaboration on development projects. With a low-code development platform, it is more straightforward to document the app creation process, providing a better transition between teams in the development projects

Development speed/Duration of Time	Development is complete within months (depending on the scale of the project)	Development can be complete within days (depending on the scale of the project)
Maintainability	Requires proper system monitoring, testing and use of third-party tools to monitor and maintain. Backend code documentation and debugging makes it difficult for a programmer to pick up another programmer's coding project.	Most LCDPs have features that support easier maintenance and monitoring with additional support from vendor.
Mobility & Collaboration	Mobility can only be achieved by leveraging a cloud-based service but collaboration requires a third-party application.	The LCDP is built on a Cloud-service platform, and leverages the power of the cloud to provide the ability for app development by multiple people in different locations on the same project.
Upgradability	Upgrade is usually done by a developer who is familiar with the system. Otherwise, proper documentation would be required for another developer to make changes to the application	Low-Code eases the upgrade process as the application can be upgraded by any low-code developer familiar with the system

CHAPTER 5: DISCUSSION

Just like DevOps, Low-Code promotes team culture, practice, and perspectives towards application development (see Figure 31). It is about multiple people (usually Business users, Developers, and IT professionals) finding better ways to work together. It bridges the gap between citizen users, citizen developers, traditional software developers who write code, and the IT group that administers the system, keeping it secure and governing the implementation lifecycle.

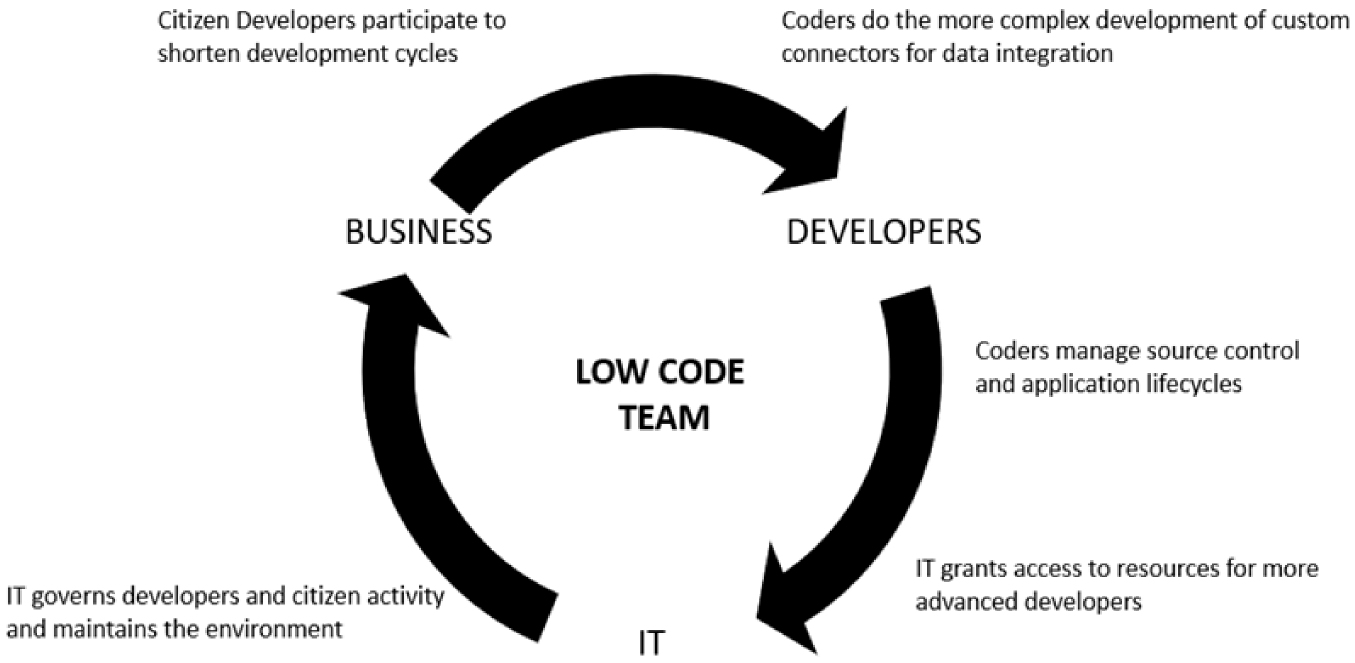

Citizen Developers participate to shorten development cycles
Coders do the more complex development of custom connectors for data integration
BUSINESS
DEVELOPERS
LOW CODE TEAM
Coders manage source control and application lifecycles
IT grants access to resources for more advanced developers
IT governs developers and citizen activity and maintains the environment
IT

Figure 31: Illustration of the Flow in a Low-Code Team Culture

CONTRIBUTIONS:

1. Approaches to Low-Code

In the Low-Code team culture, organizations can consider two main approaches to adopting a Low-Code Development Strategy:

1) The first approach is to have business users, who understand the requirements of what they need, to participate in the development process by building apps themselves. In this way business users fall into the category of "Citizen Developers". Those with some technical knowledge and interest must be offered formal training on how the LCDP works, which empowers them to join in developing applications. This approach should not exist without the IT team being empowered to properly administer software quality governance and rules, and to have systems in place to test, monitor and control application development activities. Challenges with this approach include the strain put on the IT team to control development activities, and the possibility of exhausting the organization's database resources and incurring more platform subscription costs, depending on the LCDP vendor.

2) The second approach is to have the business users working alongside developers who leverage Low-Code. With their knowledge of the bottlenecks in their business processes, developers can provide proper advice, feedback, and guidance to the IT team/citizen developers. This way, the Low-Code team can try out suggestions from business users in shorter amounts of time than with traditional approaches. Resources will be better managed, and users will be more involved in the creative process, resulting in a more regulated approach to LCDP adoption. The IT group is traditionally accountable for system quality as they are responsible for the design, build, test, deployment and maintenance of applications and they fully retain that role in this approach to Low-Code adoption.

It is important to note that, regardless of the ease and benefits of adopting Low-Code, caution must be applied with regards to the involvement of non-technical staff members throughout the application development process and lifecycle.

2. Drawbacks and Benefits of LCDP

Based on our comparison of the Low-code development approach and the traditional approach, we are able to get a deeper insight into the research questions on the drawbacks and benefits of leveraging LCDPs to rapidly scale the digital transformation of the health system.

The ability for healthcare organizations to create innovative ideas rapidly is now more important than ever for digital transformation. The Low-Code development approach offers that freedom and agility to create and innovate rapidly, thereby facilitating the evolution from digital transformation into an intelligent enterprise. The traditional approach on the other hand does not offer such development speed, so an LCDP is a big win for adopting Low-Code, especially in the context of rapid digital transformation of the healthcare system.

One of the benefits of leveraging a cloud-based platform is the power of mobility that it offers. With a cloud-based Low-Code platform, developers can create and collaborate on the same project from any part of the world. Additionally, apps produced through the platform are also accessible, given user permission at the location of the user. With the traditional approach, mobility cannot be fully achieved without hosting custom applications on a cloud-based service provider like Microsoft, Google Cloud, Amazon Web Service (AWS), IBM etc.

Although most LCDP vendors offer their products through a subscription model per user or per app depending on the service, this might be perceived cumulatively as a significant cost implication. However, compared to the cost of hiring one or more professional developers to the team with a yearly or monthly contract to build and maintain similar applications, the actual LCDP cost can be considerably lower. Vendor lock-in is still a consideration, but it might fade in comparison when considering LCDP benefits along with the perks of having product engineers supporting LCDP maintainability.

Vendor lock-in is also a factor to consider with LCDP and cloud services but this comes with both advantages and disadvantages. With regards to LCDPs in healthcare, vendor maintenance reduces the burden of system maintenance and support as the service providers are responsible for ensuring that platform and application meet specified minimal downtime and maximum uptime requirements. Maintenance response time is faster when compared to the traditional approach as the application is maintained by both internal IT staff and supported by product engineers from providers.

However, Low-code technologies come with limitations and constraints. In cases where there is the need to extend applications built with an LCDP, professional developers might have a tough time fulfilling new technical requirements by the business unit, especially requirements that can not be achieved out-of-box. In such a case the LCDP scalability might be questioned, but some service providers such as Microsoft have created a system to enable the extension of applications built on their LCDPs through the concepts of ‘*Connectors*’. Basically, they have standard and premium connectors as well as custom connectors – which enable professional developers to plug their already external application and resources (such as data from an EMR system or other in-house application) into the LDCP (see Appendix for some example connectors).

Furthermore, another major area of consideration is how the Low-Code development approach addresses concerns around security in managing healthcare data. Our findings show that the traditional approach depends on the DevOps methodology as a way of addressing security concerns during the software development lifecycle. Loopholes and bugs are addressed during the testing phase which can leave room for errors, resulting in a need to use a third-party security service. However, with the right LCDP leveraging a trusted secure cloud platform, security concerns are addressed right from the beginning of the development process based on a conditional access policy, user-based and role-based security. The concept of environmental isolation within the platform provides another security boundary at a level below the tenant-level that wraps all users, apps, data, and connections.

Overall, the aim of this thesis is to address the research question *"With the unprecedented demand on healthcare systems to optimize their internal hospital management processes due to the covid-19 pandemic, how can LCDP be securely leveraged to rapidly and optimally scale Digital Transformation in healthcare?"*. Findings from this paper show that the Low-Code development approach favourably addresses many of the criteria considered. However, as with any new technological adoption in an industry, there will be certain challenges and drawbacks to be considered. Based on our findings, we can be confident that the benefits of leveraging LCDP to rapidly scale the digital transformation of the health system outweighs any concerns we have found.

CHAPTER 6: CONCLUSIONS

There is no denying that there is a changing ecosystem in the methods, processes, types of people expertise, trends, and platforms for building apps. Traditional software development methods include approaches known to many organizations for a long time. When considering any form of technological innovation in the way software is developed, the introduction of Low-Code development has substantially brightened the vision of a rapid digital transformation in many industries.

Implications/ Contribution to Knowledge

The results from this research help to show developers and managers of healthcare technology how Low-Code helps to achieve rapid digitalization through a better understanding of drawbacks and benefits of this approach in comparison to more traditional development approaches. This can serve as a reference for hospitals to facilitate the scaling of innovative rapid development projects, using a low-code approach to develop healthcare systems. Healthcare providers and other healthcare organizations need agility to improve patient outcomes and control costs, supporting their advance into the intelligent enterprise era, especially in dealing with difficulties posed by COVID and other obstacles in the post-COVID era.

This approach also provides a more effective way of including Security right from the beginning of the development process in a security sensitive environment. Additionally, no business unit within the health system is left behind as all internal processes and workflows can be optimized, offering value across the care continuum.

Further research in the field

Findings in this paper have shown that there is considerable potential for leveraging the Low-Code approach that have not been fully utilized within the health system. This is partly due to the lack of research that explores intricate details and scenarios where low-code has been leveraged to create clinical solutions.

Further research can also be done in the application of LCDP for the development of Machine Learning and Artificial Intelligence applications within the healthcare system, particularly for the development of clinical solutions. More LCDP providers are integrating features of data-driven artificial intelligence components into their platforms and this will most likely be the development approach for many intelligent applications in the foreseeable future.

The future of software development for healthcare comes with the introduction of cloud-based platforms and Low-Code application development platforms as more people will have the agility to innovate and create applications thereby reducing the application development burden on professional programmers and increasing the rate digital innovation. Delivering enterprise-grade apps using platforms that support cloud computing to rapidly respond to change is the way to evolve into the intelligent enterprise.

Very little data have been published on Low-Code projects in healthcare and further research into various projects of this nature is needed. This would be very useful for firms and organizations who are interested in the Low-Code development approach but have no data to address their concerns about Low-code applications for medium or large sized projects.

REFERENCES

Accenture. (2018). *Cloud Native: A New Wave of Digital Disruption | Accenture*. https://www.accenture.com/_acnmedia/PDF-90/Accenture-Cloud-Native-POV-Final.pdf

Aceto, G., Persico, V., & Pescapé, A. (2020). Industry 4.0 and health: Internet of things, big data, and cloud computing for healthcare 4.0. Journal of Industrial Information Integration, 18, 100129.

Anderson, H. (2020, March 2). *Low-Code Meets DevOps: A Smart Way to Manage Modern App Delivery Demands*. DevOps.com https://devops.com/Low-Code-meets-devops-a-smart-way-to-manage-modern-app-delivery-demands/

Canada Health Infoway (2020, October 22). *Electronic Medical Record (EMR)*. https://www.infoway-inforoute.ca/en/82-our-partners/vendors/certification-services/240-emr

Crawford, A. (2021, April 6). *DevOps*. DevOps.

https://www.ibm.com/cloud/learn/devops-a-complete-guide

Dan J. (2018, February 22). *Creating an Effective App Strategy with DevOps and Low-Code.* DEVOPS Digest. https://www.devopsdigest.com/creating-an-effective-app-strategy-with-devops-and-Low-Code

Dang, L. M., Piran, M., Han, D., Min, K., & Moon, H. (2019). A survey on internet of things and cloud computing for healthcare. *Electronics*, *8*(7), 768.

Darshan D., Ramanas R., Chjaya, (2021, March 4). *Microsoft Power Platform 2021 release wave 1 plan overview - Power Platform Release Plan*. Microsoft Docs. https://docs.microsoft.com/en-us/power-platform-release-plan/2021wave1/

Evans, R. S. (2016). Electronic Health Records: Then, Now, and in the Future. *Yearbook of Medical Informatics*, *25*(S 01), S48–S61. https://doi.org/10.15265/iys-2016-s006

Frost & Sullivan (2016) Frost & Sullivan from $600 M to $6 billion, artificial intelligence systems poised for dramatic market expansion in healthcare.

https://ww2.frost.com/news/press-releases/600-m-6-billion-artificial-intelligence-systems-poised-dramatic-market-expansion-healthcare/2016

Fusion Teams: A New Model for Digital Delivery. (2019, August 2). Gartner. https://www.gartner.com/en/documents/3955908/fusion-teams-a-new-model-for-digital-delivery

Glassdoor. (2021, April 4). *Salary: Full Stack Developer*. https://www.glassdoor.ca/Salaries/full-stack-developer-salary-SRCH_KO0,20.htm

Gopal, G., Suter-Crazzolara, C., Toldo, L., & Eberhardt, W. (2019). Digital transformation in healthcare–architectures of present and future information technologies. *Clinical Chemistry and Laboratory Medicine (CCLM)*, *57*(3), 328-335. https://doi.org/10.1515/cclm-2018-0658

Greco, L., Percannella, G., Ritrovato, P., Tortorella, F., & Vento, M. (2020). Trends in IoT based solutions for health care: Moving AI to the edge. *Pattern Recognition Letters*, *135*, 346 353. https://doi.org/10.1016/j.patrec.2020.05.016

Haggerty, E. (2017). Healthcare and digital transformation. *Network Security*, *2017*(8), 7–11. https://doi.org/10.1016/s1353-4858(17)30081-8

Journal of Medical Internet Research. (2011). *Opportunities and Challenges of Cloud Computing to Improve Health Care Services*. https://www.jmir.org/2011/3/e67/

Krill, P. (2021, March 3). *Microsoft's Power Fx low-code language channels Excel*. InfoWorld. https://www.infoworld.com/article/3610390/microsofts-power-fx-low-code-language-channels-excel.html

Kumar B, B. (2020, August 25). *Low Code vs. Traditional Development*. Decode - A Publication by Zoho Creator.

https://www.zoho.com/creator/decode/low-code-vs-traditional-development

Linthicum, D. (2020, August 14). *DevOps tools best practices: A 7-step guide*. TechBeacon. https://techbeacon.com/devops/7-steps-choosing-right-devops-tools

Mell, P., & Grance, T. (2010). The NIST definition of cloud computing. Commun ACM. https://www.nist.gov/system/files/documents/itl/cloud/cloud-def-v15.pdf

Microsoft Azure (2019). What is DevOps? DevOps Explained. https://azure.microsoft.com/en-us/overview/what-is-devops/

Mohan, V., & Othmane, L. B. (2016, August). Secdevops: Is it a marketing buzzword?-mapping research on security in devops. In *2016 11th international conference on availability, reliability and security (ARES)* (pp. 542-547). IEEE. DOI: 10.1109/ARES.2016.92

MOHLTC. (2005). *Ontario's Personal Health Information Privacy Legislation for the Health Sector (Health Sector Privacy Rules) - Ministry Reports - Publications – Public Information.*

https://www.health.gov.on.ca/en/common/ministry/publications/reports/phipa/phipa_mn.aspx

Neely, S., and Stolt, S. Continuous delivery? easy! just change everything (well, maybe it is not that easy). In 2013 Agile Conference (Aug 2013), pp. 121-128.

Ness, C., & Hansen, M. E. (2019). *Potential of low-code in the healthcare sector: an exploratory study of the potential of low-code development in the healthcare sector in Norway (Master's thesis).*

https://openaccess.nhh.no/nhh-xmlui/bitstream/handle/11250/2644695/masterthesis.pdf

Ontario Hospital Association (2020). *Redesigning Care through Digital Health Implementation*. https://www.oha.com/Documents/Digital%20Health%20Ideabook_FINAL_2020.pdf

Outsystems (2019). *Application Development Trends 2019, The State of Application Development - Global Survey Report*. https://www.outsystems.com/1/state-app-development-trends/

Patel, K. K., & Patel, S. M. (2016). Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges. *International journal of engineering science and computing*, *6*(5).

Revell, M., Souther, C., Alexander, F., & Rotter, S. (2020). *What Is Low-Code? [2020 Update]*. OutSystems. https://www.outsystems.com/blog/posts/what-is-Low-Code/

Richardson, C., & Rymer, J. R. (2014). *New Development Platforms Emerge for Customer Facing Applications*. Forrester: Cambridge, *MA, USA*.

https://www.mdpi.com/2076-3417/10/1/12/pdf

Richardson, C., & Rymer, J. R. (2016). *Vendor landscape: The fractured, fertile terrain of Low Code application platforms.* https://informationsecurity.report/Resources/Whitepapers/0eb07c59-b01c-4399-9022-dfc297487060_Forrester%20Vendor%20Landscape%20The%20Fractured,%20Fertile%20Terrain.pdf

Robert M. (2021b, February 5). *Canadian Privacy Laws - Microsoft Compliance*. Microsoft Docs. https://docs.microsoft.com/en-us/compliance/regulatory/offering-canadian-privacy-laws

Rymer, J. R., Koplowitz, R., Mines, C., Sjoblom, S., & Turley, C. (2019, March). *The Forrester Wave^{TM}: Low-Code Development Platforms For AD&D Professionals, Q1 2019*. The Forrester Wave. https://wwwcdn.spanishpoint.ie/wp-content/uploads/2019/04/The-Forrester-Wave%E2%84%A2_-Low-Code-Development-Platforms-For-ADD-Professionals-Q1-2019.pdf

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020, August). Supporting the understanding and comparison of low-code development platforms. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 171-178). IEEE. DOI: 10.1109/SEAA51224.2020.00036.

Sakovich, N. (2020, September 21). *Fog Computing vs. Cloud Computing for IoT Projects*. SaM Solutions. https://www.sam-solutions.com/blog/fog-computing-vs-cloud-computing-for iot-projects/

Sánchez-Gordón, M., & Colomo-Palacios, R. (2020, June). *Security as Culture: A Systematic Literature Review of DevSecOps*. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (pp. 266-269). https://dl.acm.org/doi/abs/10.1145/3387940.3392233

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2019). Low-Code as Enabler of Digital Transformation in Manufacturing Industry. *Applied Sciences*, *10*(1), 12. https://doi.org/10.3390/app10010012

Seh, A. H., Zarour, M., Alenezi, M., Sarkar, A. K., Agrawal, A., Kumar, R., & Ahmad Khan, R. (2020). Healthcare Data Breaches: Insights and Implications. *Healthcare*, *8*(2), 133. https://doi.org/10.3390/healthcare8020133

Souther, C., Alexander, F., Rotter, S., & Alexander, F. (2019). *Low-Code vs. No-Code: What's the Real Difference*. OutSystems. https://www.outsystems.com/blog/posts/Low-Code-vs-no-code/

TELUS. (2020). *Personal Health Records | TELUS Health*.

https://www.telus.com/en/health/organizations/health-authorities-and-hospitals/patient-and-consumer-engagement-solutions/personal-health-records

The Enterprisers Project. (2016). *What is digital transformation?* https://enterprisersproject.com/what-is-digital-transformation

Tisi M., Mottu J., Kolovos D., J. de Lara, Guerra E., D. Di Ruscio, A. Pierantonio, M. Wimmer: *Lowcomote: Training the Next Generation of Experts in Scalable Low-Code Engineering Platforms, Conference of Software Technologies: Applications and Foundations (STAF 2019)*, July 15-19, Eindhoven, Netherland. https://hal.archives-ouvertes.fr/hal-02363416/document

Vincent, P., Natis, Y., Iijima, K., Wong, J., Ray, S., Jain, A., & Leow, A. (2020, September). *Magic Quadrant for Enterprise Low-Code Application Platforms* (No. G00450466). Gartner.

https://d1wqtxts1xzle7.cloudfront.net/63556609/Gartner-Low-Noo-Code-Report20200607-123018-jta9ml.pdf

Wikipedia contributors. (2020, October 19). *Software development process*. Wikipedia. https://en.wikipedia.org/wiki/Software_development_process

Zoho Creator. (2020). *What is application development?* https://www.zoho.com/creator/application-development/

GLOSSARY

Please see below the definition of some terms used in this paper.

IoT – Internet of Things:

The Internet of Things is a type of network that connects any physical object with the internet based on stipulated protocols through information sensing equipment for communication and information exchange in order to achieve smart recognition, positioning, tracking, monitoring, administration and control [31].

AI – Artificial Intelligence:

Definition from Oxford Languages, "Artificial Intelligence is the theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making and translation between languages."

ML – Machine Learning:

Definition from Oxford languages, "Machine Learning is the use and development of computer systems that are able to learn and adapt without following explicit instructions by using algorithms and statistical models to analyze and draw inferences from patterns in data."

Big data:

Definition from Oxford Languages, "Big Data is a computing term that defines extremely large data sets that may be analyzed computationally to reveal patterns, trends and associations, especially relating to human behaviour and interactions."

AD&D – Application Development & Delivery

APPENDIX

Figure 1A: Screenshot of a few of the connectors that can integrate with the Microsoft Low-code platform.

Declarative expressions and formulae

<u>Formulas:</u> You can use Power FX formulas (just like in Excel) to drive your app behavior. Just refer to controls by their name, add a dot (".") and a property name.

```
Controlname.propertyname/value
```

Note: Power FX formulas are declarative, meaning they instruct a control to listen to changes in other controls. Here we tell the value field to look out for sliders' values.

Rule(If): Just like in Excel, with the Power FX you can use If() to evaluate conditions. The basic syntax for If() is:

If(condition, value-if-true, value-if-false)

You can create formulas with If() to set properties of controls based on certain conditions.

Variables: Use variables to store information when something happens. If you want to modify a control's property while the app is running, link that property to a variable, and change the variable value with the Set() command if needed.

Variables are created automatically when you assign them a value. For example, this will set the value of the variable "`_MyVariable`" to the number 7:

```
Set(_MyVariable, 7)
```

And this will multiply _MyVariable by 2:

```
Set(_MyVariable, _MyVariable * 2)
```

View all your app's variables using View | Variables from the toolbar. Note: it's a good idea to prefix your variables with underscore ("_") - they'll be easier to recognize as variables rather than controls.

Gallery: The Gallery control is used to show records from a data source, in a scrollable list. This app includes a set of records called Contacts.

Publish: Once you save your app to the cloud and share it with people in your organization, they will be able to use it. Apps are available on the web at https://web.powerapps.com with the organization’s tenant account.