Linköping University | Department of Computer and Information Science
Bachelor thesis | Programming
Spring 2021 | LIU-IDA/LITH-EX-G--21/034--SE

Using Low-Code Platforms to Collect Patient-Generated Health Data

A Software Developer's Perspective

Agnes Hallberg

Supervisor: Peter Dalenius
Examiner: Jody Foo

Acknowledgement

Conducting this thesis has been a tedious but rewarding process whose final product would not have been possible without the continuous support of several people to whom I wish to give my thanks.

First, I would like to thank Erik Sundvall for making this thesis possible and for his continued support throughout this venture. I would also like to thank Gunnar Cedersund for letting me take part in his exciting Digital Twin project.

Continuing, I would like to thank my examiner, Jody Foo, and my supervisor, Peter Dalenius, for their helpful feedback and advice.

Finally, I would like to thank Anna Lindqvist for being a supportive friend and an endless source of inspiration.

Linköping, June 2021

Agnes Hallberg

Using Low-Code Platforms to Collect Patient-Generated Health Data: A Software Developer’s Perspective

Agnes Hallberg
Linköping University
Linköping, Sweden
agnha531@student.liu.se

ABSTRACT
The act of people collecting their health data through health apps on their smartphones is becoming increasingly popular. Still, it is difficult for healthcare providers to use this patient-generated health data since health apps cannot easily share its data with the health care providers’ Electronic Health Records (EHR). Simultaneously, it is becoming increasingly popular to use low-code platforms for software development. This thesis explored using low-code platforms to create applications intended to collect patient-generated health data and send it to EHRs by creating a web application prototype with the low-code platforms Mendix and Better EHR Studio. During the web application prototype development, the developer conducted a diary to capture their impressions of Mendix to show how a developer experiences developing in a low-code platform compared to traditional programming. The result shows that it is impractical to create applications intended to collect patient-generated health data with the two low-code platforms chosen. The analysis of the conducted diary showed that using a low-code platform is straightforward but also challenging for an experienced software developer.

Author Keywords
Low-code, Patient-generated health data, Electronic Health Records, Health apps, Citizen developer

INTRODUCTION
In today’s world, the possibilities for health data collection are seemingly endless. Transitioning from traditional health data collection by clinicians, people today are offered a vast selection of ways to collect and record their health data. Through their smartphones, people have access to thousands of applications designed specifically for health monitoring, called health apps, that can track various health metrics such as blood pressure, pulse, oxygen saturation, and glucose levels [1]. In addition, people show an increased interest in self-care, making the popularity of health apps surge [2].

Despite this, substantial barriers prevent healthcare providers and researchers from accessing and using patient-generated health data, such as a lack of integration between health apps and Electronic Health Record (EHR) used by healthcare providers [3]. Additionally, there are issues with healthcare providers using different EHR standards that lack interoperability, meaning there would be a need for integration against not only one but several EHR standards for every respective health app.

A shift is currently in progress in Europe, where healthcare providers are changing their EHRs to comply with international standards such as openEHR to increase interoperability between different EHRs [4]. This shift would allow increased information exchange between healthcare providers and researchers and simplify integration with EHRs for app developers and software providers.

Concurrently, there is also a shift in the software development world with the introduction of low-code platforms [5]. Low-code platforms are built with the concept of making coding more accessible for the inexperienced. These platforms often use visual drag-and-drop features and auto-generated code to provide an additional layer of abstraction that makes programming more intuitive.

Following this, a new kind of programmer has been created: the so-called citizen developer. Citizen developers are often domain experts with limited expertise in software development. With the introduction of low-code platforms, citizen developers can digitize and automate their work environment independently without bringing in an IT expert.

With these simultaneous shifts, the question has arisen of how these two can be combined. Can a low-code platform be used to gather the vast amount of patient-generated health data out in the world, and can a low-code platform convert the data into a format compliant with international standards for EHR systems? In addition, how user-friendly would a low-code platform be to a citizen developer, such as a healthcare provider or a patient?

AIM
The first aim of this thesis is to create a web application prototype for gathering and standardizing patient-generated health data, and by doing so, exploring the feasibility of using low-code platforms to create these types of applications.

The second aim is to assess a software developer’s opinion of low-code platforms by recording their initial impressions while working in a low-code platform.

RQ1: Which web application features intended to collect patient-generated health data can be developed with low-code platforms?

RQ2: What does a software developer experience while developing in a low-code platform?

DELIMITATIONS

Considering how many health apps are available, only Google Fit and FatSecret were selected and used during the thesis work. No consideration was given to whether the health apps were precise or accurate in their health data collection; the emphasis was put on transferring the data from the apps into the web application prototype. Likewise, any security or privacy concerns surrounding the health apps were not considered.

The thesis work only included one EHR standard and did not attempt to explore all of them.

Similarly, there are plenty of low-code platforms on the market, but only two were used for this thesis - Mendix and Better EHR Studio.

BACKGROUND

This thesis was conducted in collaboration with the Department of Biomedical Engineering of Linköping University. The web application prototype was created to be used by a research group interested in collecting patient-generated health data.

The research group was responsible for defining the requirements specification that guided the development of the web application prototype.

THEORY

This section will begin with a description of health apps and the ones used explicitly in this thesis – Google Fit and FatSecret, followed by a description of EHRs and the standard openEHR. Finally, a description of low-code platforms, including information about the platforms used explicitly for this thesis, Mendix and Better EHR Studio, as well as a brief explanation of the term “citizen developer”, will be given.

Health Apps

When speaking of health apps, it typically concerns mobile health apps, called mHealth apps for short. These apps are part of a broader concept – mHealth. The World Health Organization (WHO) defines mHealth as “medical and public health practice supported by mobile devices, such as mobile phones, patient monitoring devices, personal digital assistants (PDAs), and other wireless devices” [6]. Thus, health apps can be seen as part of a more comprehensive solution to use the ever-evolving mobile technology for healthcare purposes.

Google Fit

Google Fit is commonly associated with two different things – the health app and the open API.

Google Fitness[1] is the full name for the health app provided by Google LLC. The health app can track a user’s physical activity, steps taken and track their weight, sleep, and pulse. A significant feature of Google Fitness is its ability to receive and share data to and from other health apps, meaning many users may use Google Fitness in conjunction with other health apps.

The Google Fit open API[2] enables retrieving the various health data stored in a central repository[3] maintained by Google. The central repository stores health data from various apps, which allows the sharing of health data between these apps. If an app has permission from Google and the user, it can write and retrieve health data to and from the repository.

FatSecret

FatSecret[4] is a nutritional diary where users can log their food intake, track their weight, and export their data to Google Fit’s central repository. FatSecret was used in this thesis to collect data concerning nutritional intake. The collected data was exported to Google Fit’s central repository and pulled into the web application prototype.

Electronic Health Records

An EHR is a digital collection of medical information regarding a single patient. This includes, but is not exclusive to, things such as the patient’s demographics, history of treatment, medications, diagnosis, and lab results [7]. An EHR is a longitudinal record intended to follow and document a patient’s interactions with the healthcare system throughout the patient’s life.

Apart from storing medical information, an EHR also needs to address practicalities such as storing and transmitting the medical information securely while ensuring only authorized people can gain access to the information [8]. The goal of an EHR is to support clinicians in their day-to-day work and guarantee the highest quality of care for patients.

openEHR

openEHR[5] is an open standard providing specifications for how EHRs should store, manage, and retrieve health data. It is maintained by the non-profit organization openEHR International[6]. At the core of the standard is a two-level modeling approach, with a small, consistent reference model

[1] https://play.google.com/store/apps/details?id=com.google.android.apps.fitness

[2] https://developers.google.com/fit/overview

[3] https://developers.google.com/fit/overview#the_fitness_store

[4] https://play.google.com/store/apps/details?id=com.fatsecret.android

[5] https://openehr.org/about/what_is_openehr

[6] https://openehr.org/governance/organisational_structure

as the first level and an ever-expanding collection of clinical knowledge captured in archetypes as the second level [9].

The primary purpose of adopting two layers is to separate record-keeping concerns from clinical data, thus guaranteeing that medical providers can send and receive clinical data to each other, even as the semantics and knowledge of the clinical field keep evolving [9].

Low-Code Platforms

The term “low-code” was first used by Forrester Research back in 2014 [10]. In their report, they explain that low-code platforms are an answer to an evolving IT industry where customers are demanding faster development and deployment times. Traditional hand-coding cannot keep up with the demand, which has made several companies turn to low-code platforms as a means of speeding up development.

Low-code platforms are designed to be intuitive and easy to use to allow users to code within these platforms without the need for extensive training or education, opening up for the potential of deploying citizen developers to perform tasks previously done by an IT expert [5, 10].

The term “no-code” is sometimes used interchangeably or in tandem with “low-code”, and the distinction between the two is obscure. In general, no-code implies that no hand-coding is necessary, while low-code relies on no-code elements such as visual drag-and-drop components combined with some hand-coding.

Mendix

For this thesis, the low-code platform Mendix was chosen as the primary development environment for developing the web application prototype.

Its creators describe Mendix as a low-code platform designed for mobile- and web application development. It is intended to be used by both citizen developers and software developers. The creators of Mendix especially stress that their platform can be a bridge between these parties and allow them to work in unison through Mendix[7].

Mendix combines visual drag-and-drop elements with the possibility to customize and extend app functionality through hand-coding in Java, JavaScript, and CSS[8]. The platform is designed to support every step of app development, from the initial ideation and development to deployment and maintenance[9].

Better EHR Studio

Along with Mendix, parts of the low-code platform Better EHR Studio were used to develop the web application prototype. Better EHR Studio[10] is an openEHR-compliant low-code platform designed to be used in the healthcare sector to make health data collection and management more straightforward.

In Better EHR Studio, users can create health forms in a low-code environment where they drag and drop different widgets[11]. With these drag-and-drop widgets, a user can build various health forms adapted to different clinical settings. These health forms can then be filled in with relevant information and uploaded to a server, all within Better EHR Studio itself[12]. A feature that makes Better EHR Studio particularly interesting for this thesis is that it uploads the information saved in the health form in an openEHR-compliant format.

Citizen Developer

Baumgarten et al. [11] describe a citizen developer as a user with no previous background in IT that creates applications intended to be used either by themselves or by multiple users. Oltrogge et al. [12] use a narrower definition and coin citizen developers as people with little to no software background that develop software specifically with low-code platforms. The most defining feature of a citizen developer is their ability for software development without previous software experience, rather than what tool they use. Still, considering their lack of experience, a citizen developer often requires accessible and easy-to-use tools, such as low-code platforms, to overcome the knowledge barrier.

Citizen developers are a possible solution for the ongoing digital transformation, where industries need to adapt to an increasingly changing digital landscape simultaneously as qualified IT personnel is expensive and scarce [13].

RELATED WORKS

Most studies regarding low-code platforms are focused on exploring the market of low-code platforms [5, 10, 14] and their use for developing applications in industries other than healthcare [15, 16]. Only one study could be found where a low-code platform has been applied for healthcare purposes.

The study in question was a case study done by Totterdale et al. [17], and their aim was to use the low-code platform Mendix to aid in data collection for a research project concerning dentistry, which is similar to the aim of this study. Overall, Totterdale et al. [17] concluded that using a

[7] https://www.mendix.com/evaluation-guide/what-is-mendix/

[8] https://www.mendix.com/evaluation-guide/developing-in-mendix/

[9] https://www.mendix.com/evaluation-guide/app-lifecycle/

[10] https://tools.better.care/sandbox/studio/docs/section/getting-started/overview

[11] https://tools.better.care/sandbox/studio/docs/section/form-builder/form-builder-started

[12] https://tools.better.care/sandbox/studio/docs/section/form-builder/preview

low-code platform proved to be both practical and accessible to citizen developers, making this thesis's possible results promising.

METHODOLOGY

In this section, the theory behind the methods, autoethnography and thematic analysis, will be covered to provide the necessary context and understanding before the method section of this thesis. These methods were used to collect and analyze textual data related to the second research question.

Autoethnography

Autoethnography is a qualitative research method stemming from cultural studies, but that has begun to see application in computer sciences [18, 19]. It is a data collection method where the researcher is the participant, and the intent is to capture insider knowledge about specific experiences, lives, or relationships [20]. Autoethnography involves exploring the internal sensations and connecting them to the external world to make sense of how the self interacts with the surrounding environment.

Data is captured through autobiographical writing, where the researcher documents and problematizes events and experiences they face while being in the context of what is being studied [20]. The researcher is encouraged to write in a narrative style, where there is room for personal expression and storytelling, to involve the reader and make the research relatable to a broad audience.

Thematic Analysis

According to [21], thematic analysis is a qualitative data analysis method focusing on finding overarching themes in textual data. The thematic analysis relies on the content rather than the context of the data; it is of no relevance, for example, how a phrase was said; the emphasis is only on what was said. The thematic analysis aims to identify and describe the significant features of textual data [21].

The three steps of the thematic analysis are: (1) collecting textual data, which in this thesis will be done through autoethnography; (2) the analytic effort where the textual data is processed; and (3) the final identification of themes and subthemes within the textual data. The method used in each step will be further explained in the method section of this thesis.

METHOD

In this section, the method used to produce the results of this thesis will be described. The section is divided as to first to explain the external requirements set for the thesis and then continue to explain what methods and tools were used. Finally, there will be a description of how the collected data was analyzed.

Implementation of Web Application Prototype

Three features were attempted to be implemented in the web application prototype:

- Importing data from Google Fit through its open API.
- Administering health forms and saving the filled-in data to an EHR.
- Pre-filling fields in the health form with data collected from Google Fit.

Most of the web application prototype was developed in Mendix Studio Pro 8.18.3. Better EHR Studio 3.0.0 was used to create health forms and, combined with a small, hand-coded webserver, display and submit health forms in the web application prototype.

Selection of Low-Code Platform

This section will contain a short explanation of what motivated using Mendix and Better EHR Studio over other available low-code platforms.

Mendix

The low-code platform Mendix was chosen for a combination of reasons.

Firstly, it had been mentioned as one of the leading low-code platforms within the field [22], which gave it credibility as a dependable platform. A previous study [17] with aims similar to this thesis used Mendix with positive outcomes, indicating that favorable results could be achieved for this thesis by using Mendix.

Secondly, Mendix had a large number of accessible learning resources and documentation to assist during the development, which meant knowledge gaps would not impede the creation of the prototype.

Finally, Mendix was Turing complete, which guaranteed that all required prototype features could theoretically be implemented.

Better EHR Studio

Initially, it was planned that all development would be done in Better EHR Studio. Due to some concerns with being able to authorize against Google Fit's open API in Better EHR Studio, it was deemed safer to use Mendix as the primary development tool.

Still, Better EHR Studio was appealing to use, considering its vast capabilities for creating health forms as well as being openEHR-compliant. Thus, it was used in combination with Mendix to get the best of both worlds.

Use of Health Apps

The developer was responsible for collecting all the health data imported into the web application prototype. The health data collected were steps taken and nutritional intake in the form of calories and macronutrients. The developer collected all health data through their Android smartphone.

The developer tracked their steps taken by using Google Fitness. The steps were tracked during walks where the developer brought their phone. Google Fitness automatically exported the data to Google Fit's central repository.

FatSecret was used to collect nutritional data. The developer entered some mock data into the mobile app and gave FatSecret permission to export the collected data to Google Fit's central repository.

Collection of Textual Data with Autoethnography

During the prototype development, a diary was conducted to record the developer's impressions while working with Mendix. See Table 1 for the relevant demographics of the developer.

There were two steps to the diary. The first step was that during each workday where development was done in Mendix, the developer captured their thoughts and impressions of Mendix by narrating out loud. The narration was captured with speech-to-text software to produce notes.

Demographic	**Value**
Age	27 years
Years of programming experience	3
Previous experience with low-code platforms	None

Table 1. Demographics of the developer.

The second step was that, by the end of each workday, an entry in a diary was written where the following prompts worked as inspiration and framing device:

- What did you work on today?
- How did Mendix help you in today's work?
- How did Mendix make today's work difficult?
- What previous knowledge and experience assisted you in today's work regarding using Mendix?

During this step, the notes captured during the workday were reviewed to let the developer refresh their memory.

In combination, these two steps ensured that specific experiences were recorded as not to be lost or forgotten while also allowing for more coherent and profound reflections that come from viewing different experiences in unison.

Thematic Analysis

In this section, the process of analyzing and evaluating the collected textual data will be outlined. The section first explains how the textual data was dissected and then describes how overarching themes were discovered from the dissected textual data.

After all textual data had been collected, it was iterated through, and each sentence received a code, which essentially was one or two lines describing the content of the sentence. After the initial coding effort, the work proceeded into the next step of identifying themes and subthemes.

Themes were identified by sorting the codes. Each code was written down on a small note. The notes were then organized into different piles, where each pile represented a potential theme.

After the initial themes were established, they were then reviewed by reversing the previous sorting – codes were now attempted to be fitted into established themes to evaluate how well the themes represented the content of the textual data. Based on the result of the review, necessary adjustments were made.

As a last step, the themes were defined and labeled to distinguish them from each other. The themes were reorganized and adjusted where necessary to ensure distinct and consistent themes.

RESULT

This section will present the result of this thesis, starting with the web application prototype and finishing with the identified themes found with the thematic analysis.

Web Application Prototype

Three features were attempted to be implemented in the web application prototype:

- Importing data from Google Fit through its open API.
- Administering health forms and saving the filled-in data to an EHR.
- Pre-filling fields in the health form with data collected from Google Fit.

The first feature was successfully implemented using Mendix only. At the final stage of development, the web application prototype could import steps taken per day and the total amount of calories and macronutrients consumed per day through Google Fit's open API.

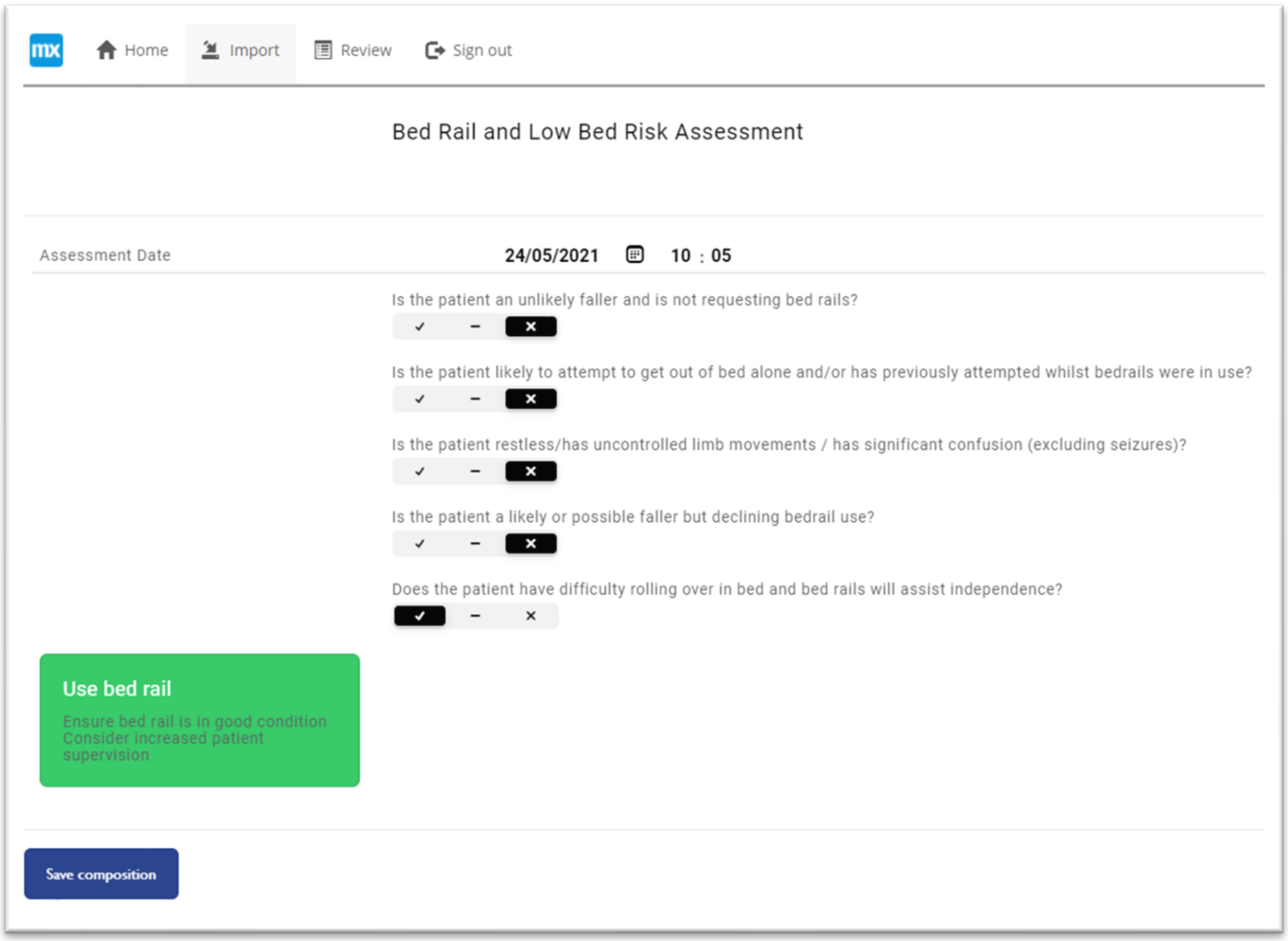
The second feature was implemented using Mendix and Better EHR Studio, along with a small, hand-coded web server. The health forms could be created in Better EHR Studio but could not be displayed in Mendix unless hosted on an independent web server. Thus, the feature could not be implemented using the selected low-code platforms solely. See Figure 2 for an illustration of a health form displayed in the web application prototype.

The third and final feature was not successfully implemented as it was deemed too technically challenging to implement with the chosen low-code platforms.

Thematic Analysis

Seven themes were identified from analyzing the textual data and were labeled as follows:

1. Quick and easy to develop in Mendix
2. Lack of functionality in Mendix
3. Needlessly challenging to implement features
4. Creativity in designing and programming
5. Lack of Mendix-specific knowledge to produce high-quality code
6. Helped by previous programming experience
7. Hampered by previous programming experience



Bed Rail and Low Bed Risk Assessment

Assessment Date 24/05/2021 10 : 05

Is the patient an unlikely faller and is not requesting bed rails? ✓ – [✓ ×]

Is the patient likely to attempt to get out of bed alone and/or has previously attempted whilst bedrails were in use? ✓ – [✓ ×]

Is the patient restless/has uncontrolled limb movements / has significant confusion (excluding seizures)? ✓ – [✓ ×]

Is the patient a likely or possible faller but declining bedrail use? ✓ – [✓ ×]

Does the patient have difficulty rolling over in bed and bed rails will assist independence? [✓ ✓] – ×

Use bed rail

Ensure bed rail is in good condition
Consider increased patient supervision

Save composition

Figure 2. A health form created in Better EHR Studio displayed in the web application prototype. The health form is filled in and ready to be saved.

1. Quick and Easy to Develop in Mendix

During the development, the developer expressed delight in Mendix speeding up the development process and providing shortcuts for implementing standard features.

"Mendix really helped in account management; there were built-in functions to authenticate users and to validate usernames and passwords. It only took a few minutes to get everything settled."

The developer also appreciated how quickly they could iterate through different solutions.

"I can iterate quickly through my solutions…"

"Mendix also allows me to experiment with the user interface quickly…"

Mendix also provided ways of automating everyday tasks while programming, such as retrieving data from API calls and converting the data into data structures within Mendix. Here, the developer expresses how JSON objects, which is a standard format for receiving data from API calls, can easily be converted to fit into data structures in Mendix:

"I really love that Mendix allows you to add a JSON snippet and it automatically translates the JSON into a data structure in Mendix."

2. Lack of Functionality in Mendix

While the developer found some aspects of Mendix to be quick and easy, others were lacking. In their diary, the developer explained that they miss certain functionality they are accustomed to from using traditional programming tools regarding handling large decision trees:

"I am having a hard time knowing how I handle larger decision trees where there is more than one variable deciding the outcome. I'd like to have other options available such as switch cases or even state machines…"

Continuous, the developer expresses how the inbuilt components and widgets of Mendix can be inflexible and difficult to adapt for particular purposes:

"I feel sometimes that the abstraction of Mendix locks me out from vital settings I need to modify to fix things…"

"Something that sucks with this solution though, is that the Iframe is "separated" from Mendix, so I can't automatically fill in fields in the form with data from Google Fit, which would have been neat."

3. Needlessly Challenging to Implement Features

Along with lacking certain functionality, the developer also expressed an issue with features being possible to implement but doing so was too challenging to warrant using a low-code environment rather than some other more traditional programming tool.

"I know where I want to go and it is a common feature, there are plenty of solutions implementing this very feature (…) I'm strongly leaning towards coding my own code snippet and putting that into Mendix rather than trying to hard-head my way to a solution through the infrastructure Mendix provides."

In this excerpt, the developer expresses frustration with authorizing against Google Fit's open API. It is a well-documented and common feature – many apps use this API after all, yet the developer struggles with its implementation in an environment advertised as user-friendly and intuitive.

4. Creativity in Designing and Programming

Another theme in the developer's diary was creativity and how Mendix both inspired and limited the creative process during the development of the web application prototype.

The developer points out that the constant visual feedback while designing the graphical user interface (GUI) inspires them and encourages them to experiment with the different GUI widgets available.

"I feel creative in Mendix, the tool itself inspires ideas with its different widgets."

"The constant visual feedback really sparks my creativity and encourages me to try out different widgets and tools just to see what I could do with them."

While inspiring, Mendix sometimes proved to be limiting. It could not provide the functionality to support all the ideas the developer wished to implement, thus halting the creative process and forcing the developer to adapt their solutions to fit Mendix.

"Creating forms for user inputs feels really non-intuitive (…) It feels narrow and inflexible."

"I want it to work a certain way, and I want to be able to modify it to meet my wants, but I feel as if Mendix won't allow it."

5. Lack of Mendix-Specific Knowledge to Produce High-Quality Code

The fifth theme entails the developer's wish to produce high-quality code combined with their uncertainty of how to do so in Mendix.

"I don't know enough about Mendix to know if my current structure will scale, but for now it works, which I have to assume is good enough."

In this excerpt, the developer expresses how they wish to structure their code to ensure that the project is scalable but cannot because they are unaware of how that is achieved in Mendix specifically. Scalability is a term often used within programming while discussing code quality. It describes how well and cost-efficient a code solution is when applied to meet smaller or bigger demands. High scalability is considered more desirable.

In another excerpt, the developer explains that their code is "messy" but are unsure if it is intolerable "messy" or within the expectations of what code looks like in Mendix:

"I still don't get the entity modeling though, it is a mess but it works. I don't know how "messy" it's supposed to get."

The developer uses "messy" in this context to explain that their code is difficult to read and understand, which links to another term commonly used while discussing code quality – readability. The readability of code is to which degree someone can read and understand that code. High readability is a characteristic often ascribed to code of high quality. When the developer writes that their code is "messy", they mean it has low readability

The two excerpts highlight the developer's wish to produce high-quality code while also showing the struggle they experience in creating high-quality code due to insufficient knowledge of Mendix.

6. Helped by Previous Programming Experience

The developer was aided by previous experience in traditional hand-coding when working with certain aspects of Mendix.

"My previous knowledge and experience of REST certainly helped understanding the steps for making a request for data from Google. "

In this excerpt, the developer divulges that making REST requests to the Google open API is similar to how it would be done with traditional hand-coding, indicating that previous experience of working with the REST protocol made the task easier.

In an additional excerpt, the developer makes another statement regarding previous programming experience assisting in the development:

"While making conditional statements I relied heavily on earlier experience. I think that at a certain point logical expressions are just something that comes to you naturally since you work so much with them as a programmer. It's just a special way of thinking."

The developer points out how creating logical expressions while programming has become second nature. They attribute this skill to their time spent programming and consider it a mark of an experienced programmer.

Overall, it would seem some skills associated with more traditional programming are transferable to Mendix.

7. Hampered by Previous Programming Experience

In contrast to the previous theme, the developer also points out how previous programming experience gave them certain expectations that proved misleading in Mendix.

"Maybe I have an issue adapting to Mendix's way of thinking around scopes because I'm used to seeing code in text rather than visual elements? I keep envisioning a text editor when thinking of scopes, like the stuff between to function brackets."

Here, the developer expresses difficulty keeping track of the scope of various data objects in Mendix and wonders if it could be due to Mendix representing code in visual elements rather than text. It should be noted that Mendix structures functions differently where a whole page is dedicated to a function, or microflow, as it is called in Mendix, rather than a section of a page. In traditional hand-coding, many functions can usually fit on one page in a text editor and can be seen simultaneously on the screen.

Due to Mendix organizing its code differently, combined with its inability to let the user view multiple components simultaneously, the developer struggles with grasping the holistic view of the code and the scopes within. It seems that Mendix requires a slightly different strategy to grasp the general outlay of the code than what the developer is used to adopting while handling traditional hand-code.

The given excerpt shows that not all expectations set by previous programming experience help with coding in Mendix. Coming in with no experience or expectations of what programming should be could, in some cases, prove to be more beneficial for understanding Mendix.

DISCUSSION

This section will contain an interpretation of the result, a criticism of the methods chosen, an evaluation of the sources cited, and a brief section outlining the ethical implications of this thesis.

Result

The result shows that it is impossible to implement all features desired for a web application intended to collect patient-generated health data with only the low-code platforms Mendix and Better EHR Studio. Only one of the three features could be implemented solely using the low-code platforms. The second feature could be implemented but required some hand-coding to do so. The final and third feature could not be implemented at all with the current solution created in the low-code platforms, as it would have required an extensive amount of hand-coding to create a solution outside the low-code platforms.

Although low-code platforms are marketed as intuitive and easy-to-use, they are still limited in what they can do. Neither Mendix nor Better EHR Studio, or the two combined, could support all the features desired for the web application prototype out of the box. Mendix's innate functionality could have been extended by integrating self-written hand-code into their low-code environment, meaning that all features theoretically could have been coded in the platform if enough effort had been made.

However, if extensive hand-coding is necessary to create the desired application, it seems more feasible to hand-code the whole application than combining hand-code with low-code. Mendix and Better EHR Studio are not flexible enough to make this kind of web application with ease, which is one of the main motivations for using low-code platforms over traditional programming tools.

Still, the thematic analysis showed that there might be some merit to low-code platforms apart from what they can achieve feature-wise. While developing in Mendix, the developer expressed feeling inspired by the platform itself and that automation of mundane tasks improved quality-of-life for them as a developer.

Even so, it was not a smooth transition from traditional programming to low-code programming.

The developer brought up several issues with developing in Mendix, such as uncertainty in how to produce high-quality code and how previous experience clashed with Mendix's workflow, indicating that previous experience with traditional programming tools may not necessarily make transitioning into low-code environments easier.

It is difficult to assess how welcoming Mendix is to experienced developers contra citizen developers, as different aspects may be of different importance for the respective groups. Experienced developers may be frustrated by the lack of flexibility in Mendix while enjoying the automation of specific mundane tasks. Citizen developers, on the other hand, might appreciate a more streamlined workflow while struggling to understand concepts in Mendix closely resembling their traditional hand-code counterparts.

In the end, it seems that Mendix was entirely accurate in their description that their platform should be used by experienced developers and citizen developers in unison, rather than just one or the other. Mendix is too simple to justify an experienced developer using it over more traditional programming tools while still being too complex for a citizen developer to manage by themselves. By working together in Mendix, a citizen developer could handle more manageable tasks and unlock time for the experienced developer to focus on the complex issues unresolvable by the citizen developer.

Method

The most significant threat to the validity of this thesis is the small participant pool with only one developer. It is hard to capture a broad picture of developing in a low-code platform as a software developer with only one developer self-reporting their experience. Similarly, only the experience of developing in one low-code platform was reported, which makes it difficult to generalize the result beyond Mendix.

Continuously, the developer only worked with Mendix for a brief period. A longer study following the developer over months rather than weeks would have allowed the developer to familiarize themselves with Mendix, resulting in further nuance in the developer's impressions of Mendix.

Furthermore, the developer was also the author of this thesis, which means the diary's content could have been influenced by the author's desires to produce a particular result. Still, the author had no personal connection with Mendix and had nothing to gain by portraying Mendix as superior or inferior, giving certainty to the author being an unbiased observer.

The reliability of the thesis is low since its result involves the opinions of only one participant. If the study were to be replicated, the result would likely vary since other participants may have differing opinions. If the participant pool had been larger, a wider span of opinions would have been captured, and the result would not be as volatile as it is now.

Also, a developer with a different programming background might have managed to implement the features of the web application prototype that this developer could not due to understanding Mendix differently or approaching the task from a different angle. A larger participant pool would have made this finding more stable as well.

Careful consideration was taken when selecting the sources cited in this thesis. An effort was made to cite reputable sources, such as peer-reviewed papers published in acknowledged scientific journals or conference papers. Still, the scientific material regarding low-code platforms was scarce, which meant non-scientific sources, such as reports produced by Forrester Research and a magazine article, were used.

Ethical Implications

Programming becoming more available to the larger masses is a question of digital democracy. If low-code platforms can do more and become more accessible to citizen developers, more people could shape the digital landscape. Still, it is essential to remember that low-code platforms are developed mainly by companies needing to make a profit, meaning low-code platforms are often locked behind paywalls. There are also concerns about low-code platforms' auto-generated code since the generated code could be of poor quality and create security risks in the developed applications.

CONCLUSIONS

The first intent of this thesis was to explore the feasibility of using low-code platforms to create web applications with the intent of collecting patient-generated health data. A web application prototype was created using two low-code platforms, Mendix and Better EHR Studio, as a method of evaluation. The result shows that the low-code platforms chosen for the thesis, Mendix, and Better EHR Studio, could not innately support all features desired for these types of applications. Additional hand-coding was necessary, which defeated the purpose of using low-code platforms over traditional programming tools.

The second intent of this thesis was to document an experienced software developer's impression of developing in a low-code platform. These impressions were collected through autoethnography and analyzed with thematic analysis. The thematic analysis showed that the developer experienced conflicting sentiments and that no conclusive answer could be given to whether developing in Mendix as a software developer is solely a positive or negative experience – it is a combination of both.

Future Work

A similar study with more participants could be conducted to establish this thesis' result further. It would also be interesting to bring in citizen developers as participants and mix experienced developers with citizen developers to see how they would cooperate within low-code environments such as Mendix.

REFERENCES

[1] W. J. Gordon, A. Landman, H. Zhang, and D. W. Bates, “Beyond validation: getting health apps into clinical practice,” *npj Digital Medicine*, vol. 3, no. 1, 2020-12-01 2020, doi: 10.1038/s41746-019-0212-z.

[2] C.-K. Kao and D. M. Liebovitz, “Consumer Mobile Health Apps: Current State, Barriers, and Future Directions,” *PM&R*, vol. 9, no. 5, pp. S106-S115, 2017-05-01 2017, doi: 10.1016/j.pmrj.2017.02.018.

[3] K. D. Mandl, J. C. Mandel, and I. S. Kohane, “Driving Innovation in Health Systems through an Apps-Based Information Economy,” (in eng), *Cell Syst*, vol. 1, no. 1, pp. 8-13, 2015/07// 2015, doi: 10.1016/j.cels.2015.05.001.

[4] C. Martínez-Costa, M. Menárguez-Tortosa, and J. T. Fernández-Breis, "An approach for the semantic interoperability of ISO EN 13606 and OpenEHR archetypes," *Journal of Biomedical Informatics*, vol. 43, no. 5, pp. 736-746, 2010/10/01/ 2010, doi: https://doi.org/10.1016/j.jbi.2010.05.013.

[5] C. Richardson and J. R. Rymer, “Vendor landscape: The fractured, fertile terrain of low-code application platforms,” Forrester, Cambridge, MA, USA, 2016.

[6] W. H. Organization, “mHealth: new horizons for health through mobile technologies,” *mHealth: new horizons for health through mobile technologies.*, 2011.

[7] R. N. Stacy, “Electronic Health Record (EHR),” ed: Salem Press, 2019.

[8] P. K. Sinha and R. J. Kate, *Electronic health record : standards, coding systems, frameworks, and infrastructures*. John Wiley and Sons, 2013.

[9] S. Garde, P. Knaup, E. J. Hovenga, and S. Heard, “Towards semantic interoperability for electronic health records,” *Methods of information in medicine*, vol. 46, no. 03, pp. 332-343, 2007.

[10] C. Richardson, J. R. Rymer, C. Mines, A. Cullen, and D. Whittaker, “New development platforms emerge for customer-facing applications,” Forrester, Cambridge, MA, USA, 2014.

[11] C. Baumgarten, A. Simeon, and M. Wilhelm, “Citizen Developers Driving the Digital Campus,” *European Journal of Higher Education IT*, vol. 2020-1, 2020.

[12] M. Oltrogge *et al.*, “The rise of the citizen developer: Assessing the security impact of online app generators,” in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018: IEEE, pp. 634-647.

[13] C. Waxer. (2020) What is Citizen Development? The Low-Code/No-Code Revolution Organizations Should Go All In On. *Forbes*. Available: https://www.forbes.com/sites/pmi/2020/12/01/what-is-citizen-development-the-low-codeno-code-revolution-organizations-should-go-all-in-on/?sh=749fd69f1da9

[14] R. Sanchis, Ó. García-Perales, F. Fraile, and R. Poler, “Low-Code as Enabler of Digital Transformation in Manufacturing Industry,” *Applied Sciences*, vol. 10, no. 1, p. 12, 2019-12-18 2019, doi: 10.3390/app10010012.

[15] R. Martins, F. Caldeira, F. Sa, M. Abbasi, and P. Martins, “An overview on how to develop a low-code application using OutSystems,” 2020: IEEE, doi: 10.1109/icstcee49637.2020.9277404.

[16] R. Waszkowski, "Low-code platform for automating business processes in manufacturing," *IFAC-PapersOnLine*, vol. 52, no. 10, pp. 376-381, 2019, doi: 10.1016/j.ifacol.2019.10.060.

[17] R. L. Totterdale, “Case Study: the Utilization of Low-Code Development Technology to Support Research Data Collection,” *Issues In Information Systems*, 2018, doi: 10.48009/2_iis_2018_132-139.

[18] S. J. Cunningham and M. Jones, “Autoethnography: A tool for practice and education,” vol. 94, pp. 1-8, 2005, doi: 10.1145/1073943.1073944.

[19] B. Chun, “Doing autoethnography of social robots: Ethnographic reflexivity in HRI,” vol. 10, pp. 228-236, 1 2019, doi: 10.1515/pjbr-2019-0019.

[20] T. E. Adams, S. L. Holman Jones, and C. Ellis, *Autoethnography* (Series in Understanding Statistics). Oxford University Press, 2015.

[21] D. Howitt, *Introduction to qualitative methods in psychology*. Harlow, England ;: Pearson, 2013.

[22] C. Richardson and J. R. Rymer, “The Forrester Wave™: Low-Code Development Platforms,” Forrester, Cambridge, MA, USA, 2016.