Norwegian School of Economics

Bergen, Fall 2019

An Inquiry into Low-Code Solutions in Institutions for Higher Education

A case study of low-code implementation at the Admissions Office at the Norwegian School of Economics

Authored by

Erik Vikebø & Ludvig B. Sydvold

Supervisor: Karen Sæbbø Osmundsen

Master of Science in Economics and Business Administration

Business Analytics and Business Analysis and Performance Management

NORWEGIAN SCHOOL OF ECONOMICS

Summary

The background for our research is the digitalisation strategy for the higher education sector 2017-2021, issued by the Norwegian Ministry of Education and Research. The strategy report addresses the potential for further efficiency and quality improvement through the utilisation of existing and new ICT solutions.

Our study is based on Design Science Research, a methodology in which one creates knowledge in the development and evaluation of an artefact. In cooperation with the Admissions Office at the Norwegian School of Economics, we have developed a low-code solution for the process of *deferment of study offer*. We have used the Business Process Model and Notation Framework to document the process and developed an artefact using the low-code development platform OutSystems. The artefact has been evaluated by industry experts to provide insight into the viability of low-code development platforms in the administration of institutions of higher education.

The artefact has demonstrated that it is possible, within a short amount of time, to develop a low-code solution that removes most manual processing steps, thus reducing processing time, while still maintaining a satisfactory overview. Our findings suggest that low-code development platforms can enable non-professional developers to contribute to application development, thus increasing the number of people that can assist in the digital transformation. However, we argue that most low-code development processes are still dependent on IT professionals running the underlying IT infrastructure.

We suggest that shared low-code development between institutions of higher education can reduce cost and development time, but differences in existing information systems and work processes might reduce the possibility of shared development among institutions.

Preface

This thesis was written as a part of the Master of Science in Economics and Business Administration, with majors in Business Analytics and Business Analysis and Performance Management, at the Norwegian School of Economics (NHH). This thesis constitutes 30 ECTS in our master's degree. Through this thesis, we have gained valuable insight into the exciting and relevant topics of digital transformation and low-code development platforms.

Many people have contributed to this thesis, which we would like to express our gratitude. We want to thank all the employees at the Admissions Office at NHH, and those of other sections who have contributed to our understanding of digital transformation in the sector of higher education.

We would also like to extend a big thanks to Avo Consulting, who provided us with advice along the way. This has enabled us to develop a low-code application and to learn first-hand about application development.

We also extend our sincerest thanks to our excellent supervisor Karen S. Osmundsen for guidance and valuable feedback.

Finally, we would like to thank our friends and families who have helped us along the way.

Enjoy the read!

Bergen, December 14th, 2019

Erik Vikebø & Ludvig Berg Sydvold

Contents

1. Introduction

The idea of being able to develop applications without writing code has been around for decades. Back in 1981, James Martin published a book called *Application Development Without Programmers*. In the preface, he writes "The number of programmers available per computer is shrinking so fast that most computers in the future must be put to work at least in part without programmers" (James Martin, 1982).

Even though there were some attempts of commercialising this idea with a so-called *fourth-generation programming language* and other *visual programming technologies* in the 1980s and 1990s, it failed to disrupt the marked (Wilfrid, 2018). Today, most analysts agree that the low-code market is on the rise (Rotter, 2019). Low-code is software that enables users to create applications through a graphical user interface instead of traditional computer programming. Forrester forecast the low-code market to represent $21,2 billion in spending by 2022, representing a compound annual growth rate of almost 50% (J. Rymer, 2018). Analysts at Gartner expect that by 2024, low-code application development will be responsible for more than 65 % of application development activity (Vincent, Lijima, Driver, Wong, & Natis, 2019).

According to the 2017-2021 digitalisation strategy from the Norwegian Ministry of Education and Research, educational institutions are expected to take advantage of the opportunities provided by digitalisation to streamline administrative support functions and ensure proper management (Kunnskapsdepartementet, 2017b). Similarly, the board at the Norwegian School of Economics (NHH) reports that their goal is to have an administration of the same quality as other leading international institutions. Processes at NHH should be effective, rational and of high quality. This will, among other things, be achieved through digitalisation (NHH, n.d.-b).

The Admissions Office at NHH is responsible for information, guidance and admissions for full-time study programs at bachelor and master level (NHH, n.d.-a). Although a strategic goal of NHH has been to make processes as efficient as possible using available technology, some processes still have the potential for becoming more efficient. *Deferment of study offer* is one of these processes. Although many elements of the process are repetitive and rule-based, the Admissions Office has not yet had the possibility to automate the process. In 2018 there was developed a Robotic Process Automation (RPA) solution for this process by a couple of students as a part of their master thesis about the viability of automating processes in the administration of higher educational institutions. Their findings suggest that RPA is a highly viable solution for administrations in higher educational institutions (Johnson & Eide, 2018). Since the RPA solution has not yet been implemented, it is interesting to study the same process using a different type of technology. This makes it possible to explore low-code implementation, as well as compare it to a technology that has already shown its plausibility.

In this thesis, the aim is to create new knowledge about the implementation of low-code in institutions for higher education, which might be extrapolated more broadly across other public institutions in Norway. The research question is, therefore, as follows:

“Are low-code development platforms a viable solution to contribute to digital transformation in the administration of higher educational institutions in Norway?”

NHH is in a position where they want to take advantage of digital technology to streamline administrative processes. NHH is not alone in wanting to do this. *Digital transformation* has been a “buzz word” in recent years for companies undergoing organisational changes related to the use of digital technology to streamline business processes (Osmundsen, Iden, & Bygstad, 2017). An inherent question is how the organisations should go about doing this. According to John Rymer, Vice President at Forester Research, companies that have started on this journey are finding that creating a digital solution for business processes is more challenging than they first anticipated. He also states that the increase in low-code development can be considered as a direct response to the pressure of digital transformation companies are experiencing today. Some of the reasons are that processes are not covered by already existing IT systems and that the development of new solutions is too slow. He also argues that low-code technology does not compete with traditional code, but can be considered a supplement, creating synergies within the organisation (J. R. Rymer, 2018).

To answer our research question, we have used Design Science Research (DSR) as our research methodology. The goal of DSR is to achieve knowledge and understanding of a problem domain by building an application of a designed artefact (Hevner, March, Park, & Ram, 2004). In the context of our research, this means developing a low-code application for the *deferment of study offer* at the Norwegian School of Economics. At the time of writing, there is little research on the viability of developing and implementing low-code solutions in institutions for higher education. Due to time-constraint and the complexity of development, we have chosen to develop a low-code application for one process only.

2. Literature

2.1 Digitalisation, Digital Transformation and Innovation

In this paper we use the definition of digital transformation given by Osmundsen et al. (2017) where *Digital transformation* is "when digitalisation and digital innovation are applied over time to enable significant changes in the way people work, leading to a significant transformation of an organisation or an entire industry". *Digitalisation* can be defined as "the process of using digital technology to change one or more socio-technical structures" while *digital innovation*, implies "combining digital technology in new ways or with physical products, to develop a new product or service which creates value for new users" (Osmundsen et al., 2017, p. 10).

Terms like *digital transformation*, *digitalisation*, and *digital innovation*, have increased in popularity among academics, newspapers, and business leaders in recent years, and Retriever is reporting that the use of *digitalisation* in articles has increased five times since 2014 (Haugnes, 2018; Osmundsen et al., 2017). The term is often used in general to talk about technology and our way of life. In business, these terms often show up in articles about companies' need to stay ahead of the curve regarding IT and efficiency. These topics are not new, but there has been a change in vocabulary. How new technology changes the way people and technology interact has been the topic of information system (IS) research for a while (Osmundsen et al., 2017). The focus of IS research can be described as the study of how information and technology are developed, implemented and is used in organisations. It also regards the consequences this has for topics like strategy, value creation, competition, structures, work processes, ways of communication, competence, decision making, planning and leadership (Osmundsen et al., 2017).

One can argue that the change of dictionary regarding the topic of information systems in business is a result of the technological development over the last 50 years. According to Porter and Heppelmann (2014), IT has gone through three phases which have been the driving force of business development and society in general. Phase one was in the 60s and 70s. Back then, IT helped automate simple tasks in companies and the value chain. Phase two, in the 80s and 90s, was brought about by the rise of the internet. Inexpensive connectivity enabled coordination and integration across individual activities, suppliers, customers and geography. The third phase is the one we are in today. IT is now an integrated part of the product or service itself. We see this in, for example, small electronic appliances in our daily lives. In business, cloud services, sensors, and big data are major topics (Porter & Heppelmann, 2014), and IT has become a transparent part of the organisation, product, service, as well as society (Osmundsen et al., 2017).

In the definition of *digitalisation*, we use the expression *socio-technical structures*. This means that *digitalisation* is more than just the use of digital technology. This is called *digitisation*, which is the process of making analogue information into digital (Yoo, Henfridsson, & Lyytinen, 2010). *Digitalisation* affects the way humans and technology interact. Socio-technical structures can be made up of two joined, but independent systems, a technical, and a social. The first is made up of processes, tasks, and the technology needed to transform input to output, while the latter consists of the attributes of people, like attitudes, skills, values, relationships and authority structure (Bostrom & Heinen, 1977). When automating the process of *deferment of study offer* at NHH, we are not only changing a technical aspect of a process. We are changing the way people and technology interact to get work done.

Digital innovation is defined by some researchers as a process, while others focus on the result (Osmundsen et al., 2017). As a result, digital innovation can be defined as “A new product or service which create new value for the adopter, developed by combining digital technology in new ways or with physical components”. As a process, digital innovation can be defined as “to combine digital technology in new ways with physical products, to develop a new product or service which creates value for the adopter” (Osmundsen et al., 2017). Although automating a process in an admissions office does not sound like innovation, it can be called innovation in the eyes of the adopter. Furthermore, when *digitalisation* and *digital innovation* gets to affect an organisation over time, this can lead to a significant transformation of an organisation, and sometimes a whole industry (Osmundsen et al., 2017). In the context of NHH, when processes are automated, with digital technology in new ways, this can digitally transform the organisation into a more efficient institution.

2.2 Information Systems

To understand the challenges with digital transformation, one must also understand what information systems (IS) are. IS can be defined as an organised combination of people, hardware, software, networks, data resources, and policies. It also covers the procedures that stores, retrieves, transforms, and disseminates information in an organisation (Urquhart &

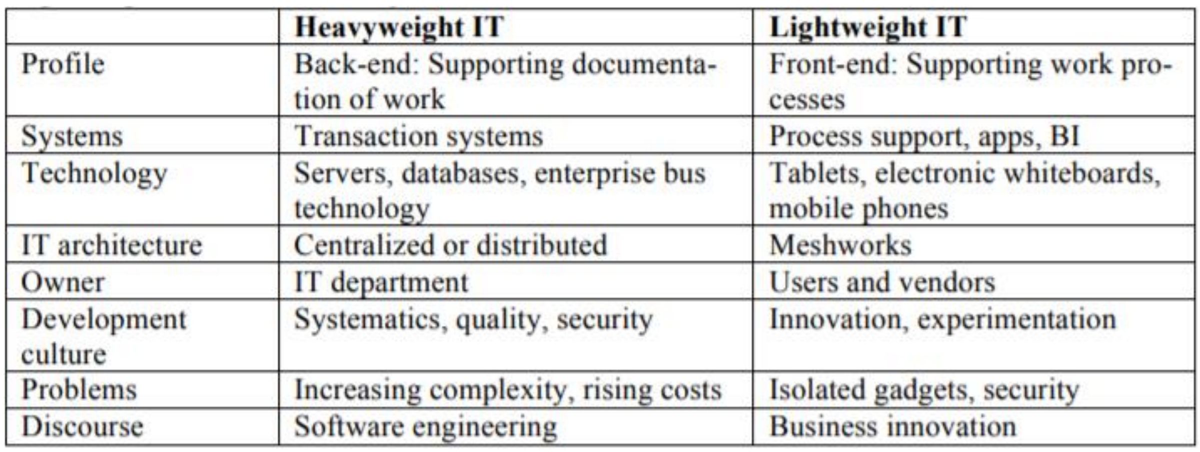
Ravindranathan, 2006). IS supports business processes, operations, decision making, and competitive advantages (Urquhart & Ravindranathan, 2006). It does not, however, have to include computers and other information technology (IT) components, but in this paper, we exclusively talk about IT-supported information systems (Heggernes, 2017). When we talk about IT, we are only talking about the technical aspects of information systems. The IT components in an information system can be integrated with each other to varying degrees. For example, at one end, you could have an information system which contains a process where a human uses a calculator, as a part of their work routine as a cashier. On the other hand, you have technology components that are seamlessly integrated with the rest of the IS, for example, automated cash registers that are increasingly being used in grocery stores all around Norway.

In the 2015 article, *The Coming of Light Weight IT*, Bygstad points out that the IT architecture in today's organisations is experiencing two different developments. On the one hand, organisations are becoming increasingly digitally integrated. We see, for example, an increased dependency on the internet as companies are shifting from on-premise solutions (local data storage) to cloud-based data storage. These cloud services are also used to run the software as web services. There is also an increase in the software's capabilities in communicating with each other, for example, through application programming interfaces (API). This has led to further integration of information technologies in information systems (Bygstad, 2015). This development has decreased the work process complexity of IS systems because many processes in the IS can be automated by integrating IT software. As IT architecture is becoming more integrated, the complexity, and cost of maintenance is increasing, because one component has the potential to affect all the others (Sommerville et al., 2012).

The second development concerns people using more digital technology in their daily lives (Bygstad, 2015). IT is no longer only the domain of the IT department, as technology is often used both at home and at work. Since the technology that employees use, such as smartphones, often do not belong to the company, it may be considered as a security risk. However, it also opens new possibilities as a whole industry of software development has been created for these technologies.

Bygstad et al. (2015) propose two separate socio-technical knowledge regimes, *heavyweight* and *lightweight IT*. *Heavyweight IT* concerns the IT service we get from IT departments today.

It is characterised, among other things, by being focused on back-end functionality and having an IT architecture that is centralised and distributed. The development culture in these departments can be generalised as being focused on systematics, quality and security. The challenges this focus brings is that development and maintenance are complex, and that cost is high. *Lightweight IT, on the other hand,* focuses on front-end processes, and a lot of the development is based around a network, for example, social platforms. Development culture is more innovative and experimental, but the technology is often isolated from the rest of the IS and is often associated with security concerns.



	Heavyweight IT	**Lightweight IT**
Profile	Back-end: Supporting documentation of work	Front-end: Supporting work processes
Systems	Transaction systems	Process support, apps, BI
Technology	Servers, databases, enterprise bus technology	Tablets, electronic whiteboards, mobile phones
IT architecture	Centralized or distributed	Meshworks
Owner	IT department	Users and vendors
Development culture	Systematics, quality, security	Innovation, experimentation
Problems	Increasing complexity, rising costs	Isolated gadgets, security
Discourse	Software engineering	Business innovation

Figure 1: Heavyweight and lightweight IT (Bygstad, 2015)

Low-code platforms, described in more detail in subsection 2.3, are well suited for the kind of development which is characterised as lightweight IT. It can work as a part of the front-end systems supporting processes, creating interfaces which support business intelligence and is well suited for experimentation because of the quick development. It is, however, also well suited as a part of a heavyweight IT development. As a result, low-code development platforms are not restricted to either one of these categories. This aspect separates it from RPA solutions which are considered to be non-intrusive, and exclusively lightweight (Bygstad, 2015). Bygstad suggests that when lightweight development is to be implemented in an organisation, in their case RPA, it should not be done as a part of the department that works with heavyweight IT. The development should rather be organised outside the IT department, or as a separate team within the department. This is because of the different development culture, shown in *figure 1,* as well as the lack of resources and time which is often experienced in IT departments (Bygstad, Stople, Steinsund, & Iden, 2017). Since low-code is more complex and intrusive, the IT department will still be needed to create and support the underlying IT infrastructure (Everhard, 2019).

2.3 Low-Code Development Platforms

In this section, we address the concept of low-code development platforms, hereby referred to as low-code. In software development, speed is essential. Consultant and expert on software delivery Daniel Terhorst-North write that "the goal of software delivery is to minimise the lead time to business impact. Everything else is detail" (Terhorst-North, 2013, p. 1). Lead time is the time between the software is being requested, and the moment the software is delivered to the customer (Konschake, 2018). The challenge is that programming new applications using traditional code languages like Python, JavaScript and C++ is a time-consuming and labour-intensive process. Further, making changes to a system with tens of thousands of lines of code can be cumbersome and slow (Richardson & Rymer, 2016b). Therefore, organisations and application development teams are looking to adopt any techniques that will accelerate software development and delivery (Marvin, 2014). This is where the benefits of low-code come into play.

Low-code is not an entirely new concept, but rather a practice that had never been defined before Forrester coined the term back in 2014. Forrester defines low-code as "platforms that enable rapid application delivery with a minimum of hand-coding, and quick setup and deployment, for systems of engagement" (Richardson & Rymer, 2014).

By being platforms, low-code enables the developer to create applications for a lot of different scenarios for both web and mobile devices. This can, for example, be applications that enhance customer experience, improve business operations or modernise legacy systems. Even though it is possible to develop applications without writing a single line of code, low-code enables the developer to extend applications with custom code.

When researching the low-code market, analysts at Forrester interviewed dozens of companies that were using low-code to speed development. In their report, they concluded that faster delivery is the primary benefit of utilising low-code. They estimate that low-code accelerates application development by five to ten times compared to traditional application development. However, they also found that low-code help organisations respond more quickly to feedback after initial software releases (Richardson & Rymer, 2014). Even though Forrester's definition is concise, we believe that low-code expert and writer Matthew Revell, gives the reader a better understanding of the concept:

> *“Low-code describes a family of tools that helps developers create complete applications visually using a drag-and-drop interface. Rather than writing thousands of lines of complex code and syntax, low-code platforms allow users to build complete applications with modern user interfaces, integrations, data and logic quickly and visually”* (Revell, 2019, p. 1).

When Revell mentions a family of tools, he refers to the fact that there is no single low-code technology, but rather a market with over 100 vendors (Taulli, 2019). A drag-and-drop interface implies that the user can select a virtual object and drag it into a preferred location. The technique is considered easier to learn compared to traditional code and syntax (Feldman, 2013). However, writing code is not the only complex and labour-intensive part of application development. Planning, testing, and implementation can be just as time-consuming. Modern low-code development platforms need to simplify the entire application development lifecycle. This includes debugging, integration, performance analysis, and deployment (Ross, 2018).

2.4 Low-Code Compared to Robotic Process Automation

Low-code development platforms are sometimes compared or even confused with other drag-and-drop tools, such as Robotic Process Automation (RPA). Even though many low-code and RPA-projects have the same goal of streamlining administrative processes, their approaches are quite different. By interacting with the user interface and locating data fields through the code of web pages and underlying systems, RPA mimics digital tasks that are usually performed by a human (Johnson & Eide, 2018). In contrast, low-code is built to quickly deliver new applications, either as a supplement to existing systems or as a complete replacement. When integrating with existing systems, low-code connects through an application programming interface (API) (Murphy, 2018). APIs allow developers to access, update or delete information on a remote computer without having to understand the technical details of the system they are interacting with (Braunstein, 2018). In simple terms, an API act as a middleman, taking the request from one piece of software and then replying with the appropriate response from the other (Tanna, 2016). This means that low-code, unlike RPA, is not fragile to changes in the user interface. Also, if the application is lacking necessary features, RPA will not address them, as it only uses the application in its current state. In those situations, low-code could be a better choice. However, RPA is a great tool for quickly automating a process where existing systems are difficult and costly to replace, and there is no option for integrating to them via modern APIs (Murphy, 2018). In certain cases, the two technologies can also complement each other, where the RPA works as an integration between the low-code platform and existing infrastructure (Kovalev, 2018).

2.5 Preparation and Implementation of Low-Code

An advantage of low-code, compared to traditional software development, is that it is more accessible for people with a non-technical background, often referred to as citizen developers. Gartner defines a *citizen developer* as a "user who creates new business applications for consumption by others using development and runtime environments sanctioned by corporate IT" (Gartner, n.d.). The increase in people that can assist in the development opens many opportunities, as well as potential pitfalls. In a 2019 article, Senior manager at KPMG Digital Sebastiaan Tiemens lists five steps to prepare businesses for low-code implementation (Tiemens, 2019):

1. Discover

When introducing low-code within an organisation, it is important to demonstrate its capabilities and strengths right away. One of the advantages of low-code is that functional Proof of Concepts can be developed within a short amount of time. This enables all types of organisations to quickly start experimenting with the technology. It is recommended to start with minor projects and not spend too much effort on administrative processes and formalisation.

2. Vision

At this stage, it is time to start structuring initiatives and develop a low-code vision. While developing the vision, it is important to take a broad view and include topics such as digital business strategy, sector developments and other emerging technologies. It is recommended to keep it simple and create a compelling low-code story which will gather awareness and traction across stakeholders. It is also recommended to decide on a low-code platform that fits the need of the organisation.

3. Sketch and mobilise

After experimenting with the technology and developing a low-code vision, it is time to transform the vision into a formalised plan and start piloting. The pilot can be run in a certain business unit or organisation-wide. Before implementation, the organisation must develop a business case. The goal of developing a business case is not to calculate the exact value of the investment, but rather to demonstrate that the investment is reasonable (Christensen, 2018). At this stage, the organisation will learn what the value options are and how the technology will impact their operating model. A functional design, attention to technical aspects as coherent logic, and architecture are essential for a successful implementation.

4. Launch and realise

Even though low-code makes development, integration, performance analysis, debugging and deployment easier, it is still necessary to organise and manage the project like a traditional software implementation. Low-code enables business and IT to unite and create mutual understanding, support Agile development and increase the likelihood of a successful implementation. Still, care for software quality and security, life cycle management and maintenance processes are crucial to be in control and manage quality and costs in the long term.

5. Scale and improve

Leveraging low-code at scale will increase the organisation's capabilities to adapt to new circumstances. Combining business knowledge with technical capabilities will help the organisation determine the best way forward and establish a plan for upscaling. When the organisation is leveraging low-code at scale, it is recommended to establish a Centre of Excellence. Gartner defines *Centre of Excellence* as "a physical or virtual centre of knowledge concentrating existing expertise and resources in a discipline or capability to attain and sustain world-class performance and value" (Pemberton, 2016). The Centre of Excellence will ensure coordination, alignment, consistency and continuous improvement.

3. Digitalisation in Norwegian Higher Educational Sector

As a public Norwegian business school, NHH is obligated to follow the Ministry of Education and Research's strategy guidelines. In recent years, the Ministry of Education and Research has made it clear that the higher education sector must do more with fewer resources, and they recognise that technology can play an important part in making the sector more efficient. In 2017, the ministry published the report *Digitaliseringsstrategi for Universitets- og Høyskolesektoren* regarding the digitalisation strategy for the Norwegian higher educational sector (Kunnskapsdepartementet, 2017a). Among several topics in the report, digitalisation of support functions is mentioned. The idea is that if institutions digitalise and automate support functions, the institutions will gain a higher capacity for their core activities, which are research, innovation, teaching and communication.

Since most higher education institutions in Norway are public, including NHH, there is not much competition between institutions. This has made it possible to make strategies on a more aggregate level. The vision described in the report is one where institutions solve as many of the administrative support processes as possible in unison. By doing this, they want to find synergies between the institutions, despite their differences and find appropriate user-friendly solutions for students and employees. The ministry has requested institutions to include digitalisation in both the planning and the design of the work processes. They also have more specific suggestions regarding which systems should be used. For example, the archive system P360 used at NHH is one of two archive systems suggested in the report. They also have standards for how data in the information system should be handled, which is always to be available digitally and stored in one location. From this location, it should be distributed to everyone who needs it.

Educational institutions have done a lot to digitalise their services. For example, most schools in Norway has the possibility to film lectures and make them available on their webpages. In general, one can say that teaching in this sector is very close to being paperless. There has also been some cooperation between institutions of higher education in Norway to create common digital solutions. Examples include *Samordnet Opptak* (SO) which is a common portal for admission applications where the applicant only needs to submit one application, which is then distributed to the schools and universities the student wants to attend. If the applicant has good enough grades, he or she will get an offer from the institution which was ranked the highest by the applicant (Samordnet Opptak, 2008). *Fellessystemet* (FS) is another system which follows all students through their higher education. FS stores information about each student’s subjects, study program, and degrees, and shares this information with other institutions (Unit, n.d.). Apart from these achievements, it is also mentioned in the Ministry of Education and Research’s strategy report for digitalisation that the options to integrate with some of the sector-wide systems has been lacking. The API for FS launched as late as 2018. A consequence of this has been that data is often stored in multiple locations.

Another aspect that is brought up in the strategy report is the cost reduction that can be achieved if organisations stay away from customised IS solutions. For example, it is preferred that institutions decide to share digital solutions and implement solutions that are regarded as best practice in the market. This will reduce the need for external consultants when implementing, upgrading or maintaining the system. It would also be easier to integrate with other institutions at a later point in time. The main principle of financing in the sector is that every institution is responsible for its own costs. It is also their responsibility to choose the IT systems which are needed for the processes at their institutions. Every institution shall steer their digitalisation effort through their own strategic goals, which should accommodate the digitalisation strategies published by the Ministry of Education and Research.

4. Method

4.1 Design Science Research

Design means “to invent or bring into being” (Vaishnavi, Kuechler, & Petter, 2019, p. 3), and *design science* is a broad category of research about design as a concept, as well as research on the design of objects. What separates design science from design science research (DSR) is the creation of an artefact as the means of acquiring knowledge (Vaishnavi et al., 2019).

An *artefact* is an “artificial object”, which should be understood in its most neutral sense, meaning, man-made as opposed to natural (Simon, 1970). A characteristic of these artefacts is that they are created to meet a goal. Simon (1970) divides the design of an artefact into three aspects, the *inner environment*, which can be thought of as the technical aspects of the *artefact*, the *outer environment*, which is its surrounding environment, and the *interface* between the two which is the solution to meet the goal. An essential aspect of these artefacts is that they continuously need to adapt to the *outer environment* if they are going to continue to work as an *interface* between the two environments. To clarify the terms above, we can put them in the context of the application that was designed for this paper. The *inner environment* is the technical elements which our application is made up of. This includes subsection *2.3 Low-code Development Platforms*, and the design process which will be further described in chapter *6. Development.* The *outer environment* is everything that our application must relate to; that includes chapter *3. Digitalisation in Norwegian Higher Educational System*, and how information systems have evolved, subsection *2.2 Information Systems*. The *artefact* is, therefore, the *interface,* between the designed application, and the function it is intended to fulfil at NHH.

The general idea of the DSR method is that the researcher can gain knowledge through the design of an artefact. By evaluating the artefact, we will get a better understanding of the problem in which we are trying to solve. In general, DSR consists of two processes, build and evaluate, which will be iterated several times. This is done to further improve the quality of the artefact and the design process, and enhance the knowledge of the environment under scrutiny (Hevner et al., 2004).

4.1.1 DSR Process Model

The framework we will use in this thesis is the DSR process model described by Vaishnavi, Kuechler, and Petter, (2019). The model is an adaption of a model initially developed by Takade et al. (1990) for computable design process models, but the model also fits well for DSR, although the content in each step is different (Vaishnavi et al., 2019).

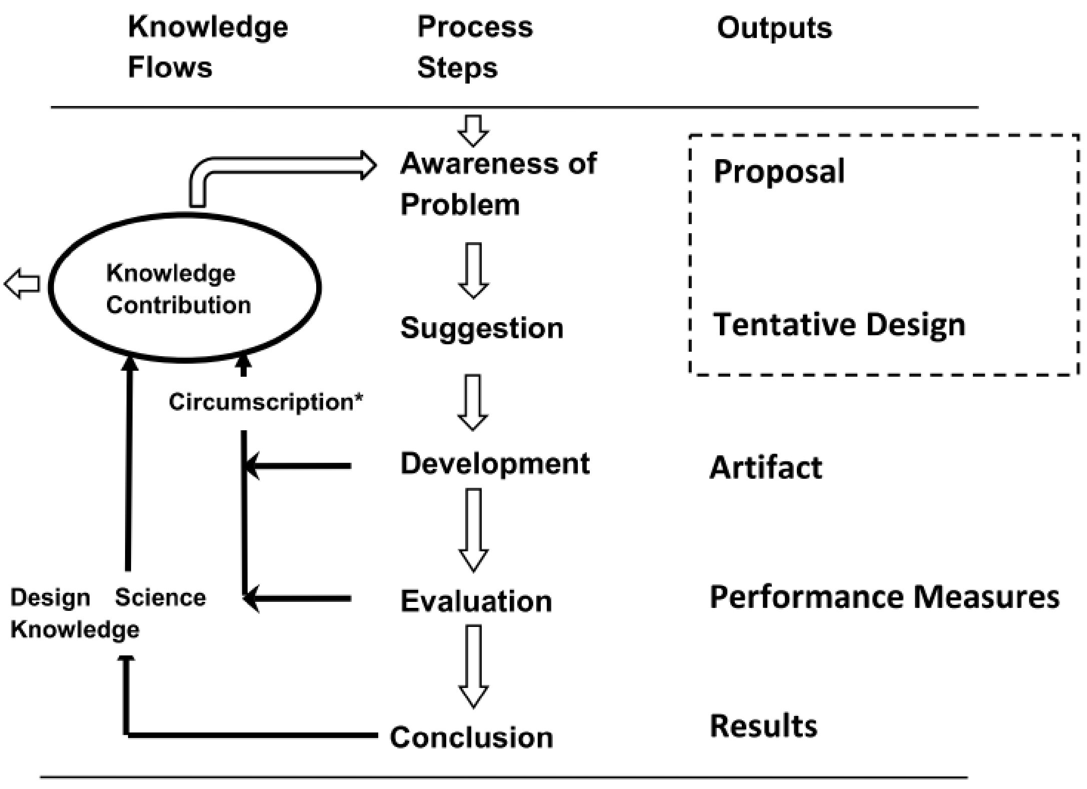


Figure 2: typical design science research effort (Vaishnavi et al., 2019)

As *figure 2* illustrates, the DSR process is divided into five steps, that can be iterated through multiple times. Each process step creates output for the research process. Usually, going through these steps once is not enough to create a satisfying *interface* between the environments, and it is common to iterate through several times before concluding. In *figure 2,* this iteration is called *circumscription.* In the end, we hope to gain *design science knowledge* that can contribute to the research field of digital transformation.

1. Awareness of Problem

In the first step of the model, awareness of the *outer environment* is created. In our case, the outer environment is characterised by the Admissions Office's, NHH's, and the Norwegian government's strategic demand for increased efficiency, the knowledge of the people at the

Admissions Office, and the information system the process is a part of. When we contacted the Admissions Office at NHH, we were presented with several manual processes that the employees at the office felt were cumbersome. The Admissions Office could not do anything about these processes because the systems they were using were not integrated, and they did not have the tools or knowledge to do so. We chose the process of *deferment of study offer* for our study because the process fit the timeframe we needed for development. It was also a compelling case because an RPA solution had already been made. Developing another solution with different technology made it possible to compare the two approaches. To gain knowledge of the *inner environment* of the process we used *Business Process Model and Notation* (BPMN). BPMN provides a graphical notation for expressing business processes. The objective of BPMN is to support business process management for both technical and business users by providing a notation that is intuitive yet able to represent the complexity inherent to business processes (White, 2004). Through our development, our goal has been to create an artefact that makes the best fitting interface between the two environments.

2. Suggestion

The second step presents a proposal for an *artefact* that would be the solution to the problem described in *1. Awareness of problem,* rearranging existing elements or introducing new elements. It is typical for this suggestion to present a tentative design or a prototype. This prototype does not need to be completely functional, but it creates a roadmap, which shows what kind of artefact the researcher wants to create (Vaishnavi et al., 2019). We will propose a solution by visualising it using the same BPMN framework used in *1. Awareness of Problem*. This makes it easier to compare the new and the former process solutions. We will in this paper, propose a low-code solution to make the process more efficient.

3. Development

In the development stage, an attempt is made to develop the proposal from step 2 into a functional artefact (Vaishnavi et al., 2019, p. 12)**.** We will create an application in cooperation with NHH to automate the process in question. Most of this development will be based on educational videos about the low-code development platform OutSystems, which is one of the world’s most prominent low-code vendors. We have also cooperated with Avo Consulting, who use this platform to develop digital solutions for some of their clients in Norway. The arrangement was that we could ask them about the OutSystems platform if we got stuck during development. The development will take place over two periods, which are called *sprints*. The content of these sprints will be explained in chapter *6. Development*.

4. Evaluation

An artefact is complete and effective when it fulfils the requirements and satisfies the constraints of the problem it was meant to solve (Hevner et al., 2004). We emphasise here that this does not mean that the *interface* between the *inner environment* and the *outer environment* must be fully functional, but the design of the artefact must be good enough so that the *artefact* can be evaluated and that knowledge can be produced in the process of making the *artefact*.

According to *Design Science in IS Research* (Hevner et al., 2004), an artefacts' utility, quality, and efficiency must be rigorously demonstrated through evaluation methods. These tests must be based upon the environment in which the artefact is to operate. In the case of IT artefacts, the evaluation must include the integration within the technical infrastructure, as well as the business environment. The evaluation of *artefacts* should ideally be done with objective criteria. If someone developed IT software as an *artefact*, one could run stress tests to see whether the design would handle extensive use. As will be shown later, we were not able to create a fully functional *artefact*, and therefore could not test it in any quantitative way. Consequently, we were not able to measure the efficiency of our *artefact*. We have however performed a qualitative evaluation through interviews to evaluate the *artefact's* utility, as well as quality.

5. Conclusion

In this phase, the result of the research effort is summarised. Although developing an application is not the primary goal of the project, our experiences developing an application for an organisation in the educational sector may be replicated by others who want to implement similar technologies. The primary purpose of the conclusion is to make a strong case for the knowledge contribution to the field of research (Vaishnavi et al., 2019). In our case, we hope to contribute to the research on digital transformation in the sector for higher education in Norway.

4.2 Interviews

To support the DSR approach, we have conducted interviews with the Admissions Office and Avo Consulting. The interviews have been used to gain knowledge about the current process for *deferment of study offer,* expert advice regarding some aspects of development, as well as evaluating the finished artefact. The interviews helped us to investigate whether the artefact was successful as an interface between the outer and the inner environment at the Admissions Office. To conduct the interviews, we used a semi-structured interview approach. This is a qualitative method of interviewing, characterised by open-ended questions which are asked in such a way that invites the informants to share their thoughts about the subject at hand (DiCicco-Bloom & Crabtree, 2006).

Finding key informants was essential. We decided to perform several interviews to evaluate our artefact. To gain in-depth knowledge of the work process in which we were trying to improve, we performed interviews with several employees at the Admissions Office. The employees at the Admissions Office were also in a good position to evaluate whether our artefact would be suitable to replace the old solution.

To obtain valuable insight into low-code and application development we interviewed consultants at Avo Consulting. The consultants had experience in implementing low-code solutions for both Norwegian and international customers. We thought their knowledge would give us valuable insight into how application development is conducted in real life. Before the interviews, we created interview guides to make sure that we covered the topics we found relevant.

To document the interviews, we used audio recordings. The interviews were conducted in Norwegian before being translated to English. The tapes were deleted after the transcription was completed. Each informant received an information letter stating the intentions of the interview. They also signed a declaration where they accepted the interview session to be recorded, transcribed, and used in our research. After the interview, these transcriptions were made available to the informants so that they could review them if they wanted. To prepare the data for analysis, we sorted the responses into topics for later use.

4.3 Agile Software Development

When developing our artefact, we have chosen to use an Agile approach. *Agile software development* refers to “a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organising cross-functional teams” (ISTQB, n.d.). The idea underlying an agile approach is that the client hardly has the prerequisites to define all requirements before the project start-up. The client's requirements and the solution's possibilities should challenge each other through a close dialogue between the client and the supplier (Christensen, 2018).

In the development phase, we have used the SCRUM method, which implies that we have organised the development in several sprints that build up to a release. Each sprint has a very schematic and strictly regulated schedule. The original design can be challenged and further elaborated if deemed necessary. One can also re-prioritise, meaning that an item determined for a later sprint can swap place with an item from an earlier sprint. Items can be withdrawn or postponed for a later sprint. In short, it is an agile approach to development, where the client plays a significant role. By definition, a sprint cannot be delayed because the timeline is fixed. However, a sprint may have remaining items when the sprint is completed. Remaining items must be developed at a later stage, possibly by adding a new sprint (Christensen, 2018). Ultimately, an acceptance test is conducted to evaluate if the system is in compliance with the business requirements and whether it is ready for implementation (ISTQB, n.d.).

Agile methods are all about embracing the fact that a requirement specification hardly reflects the only true path towards the goal. We must understand that when interacting in a development process, new ideas will be born that sometimes exceed what has been proposed in a previous phase. Agility is about the ability to incorporate these suggestions along the way (Christensen, 2018).

5. Business Process Documentation

In this chapter, we will present the business process documentation. By presenting the current process and system landscape, we will try to create awareness of the problem. This includes an introduction to the system landscape, how the process of *deferment of study offer* is handled today, and how a low-code application could change this process.

5.1 Current Process for *Deferment of Study Offer*

The regulation of full-time studies at NHH states that study offer can be deferred by up to two years if applicable. This implies that students who for some reason, need to postpone their studies at NHH can do so for up to two years. Typical reasons for deferment are compulsory military service, pregnancy, adoption, and the like. We will start by giving a walkthrough of the current process as well as an introduction to the system landscape. A BPMN mapping of today's process is illustrated in *figure 4*.

To be entitled deferment of study, applicants must accept the study offer and then apply for deferment within given deadlines. Today, the process of *deferment of study offer* at NHH includes two roles: the applicant and a part-time employee processing the applications. The part-time employee is hereby referred to as a caseworker.

System landscape

The system landscape consists of a Web form, Microsoft Outlook, Microsoft Excel, a local directory/server, the common student system (Felles Studentsystem, FS) and the archive system Public 360. None of these systems directly integrate with each other, which implies that the information needs to be transferred manually between the systems. An overview of the system landscape is given in *figure 3*.

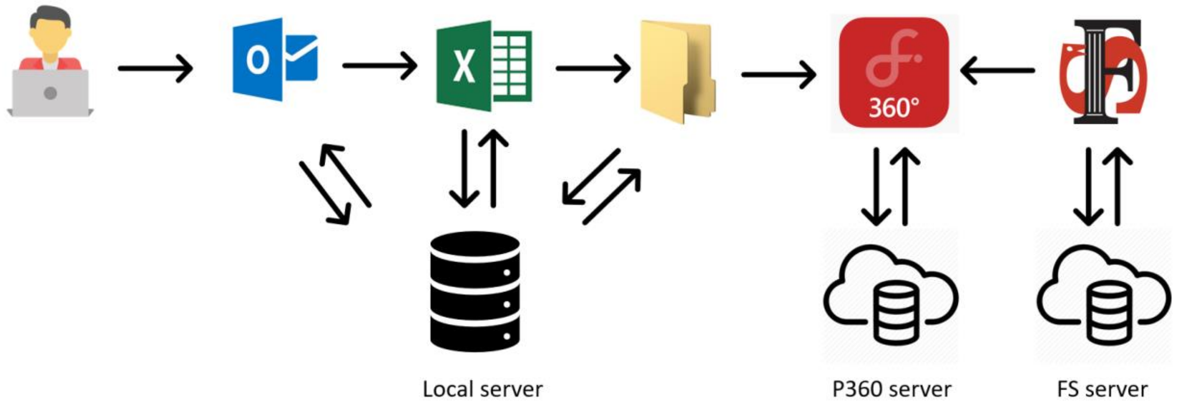


Figure 3: System landscape

Web Form, Outlook and Microsoft Excel

The process starts when a student applies through a web form at nhh.no. The content of the web form is edited in collaboration with the Admissions office, while NHH's IT-department maintains the webpage. The application must include application number, study program, name, address, email address, phone, cause and years of deferment. Applicants are also asked to upload a document that proves that they are entitled to receive deferment. If the document contains sensitive information, it shall not be uploaded, but rather be sent by post. The reason for this is that the application is received in an Outlook inbox which is not approved for storing sensitive information. If sensitive information is received in Outlook, it will be processed and then deleted afterwards.

After the application is received, a caseworker at the Admissions Office manually transfers the application to a dedicated inbox in Outlook. Next, the caseworker creates a new folder where the application and attachment are saved. The caseworker then opens an Excel spreadsheet and inserts application number, date, name, cause and years of deferment. While Outlook and Excel are licenced through Microsoft's Office package, the data is stored on a local server owned by NHH. This server also contains the folders where the applications are stored before they are uploaded to the archive system.

Public 360 and the Common Student System (FS)

After updating the spreadsheet, the caseworker creates a new case in Public 360 (P360). P360 is a cloud-based archive system licenced by NHH, through the Finnish IT-company Tieto. P360 is widely used throughout higher education institutions in Scandinavia and is approved to store sensitive information. Depending on the content, the caseworker marks the case as either sensitive or ungraded. Because the applicant does not currently study at NHH, the caseworker must create a new student in P360. This is done by inserting the information provided in the application form. For Norwegian applicants, the caseworker also inserts the national identity number. This is accessed through FS, which is an administrative system for Norwegian universities and colleges. FS is developed and managed by Unit - the Directorate for ICT and joint services in higher education and research (Unit, n.d.). FS is not approved to archive sensitive information, herby the need for P360. By inserting the Norwegian identity number, the caseworker enables an option to send a response via the digital mailing service Digipost. After the creation of a new student and case, the caseworker uploads the application and attachment to P360.

To document that the application is uploaded, the caseworker opens Excel and changes the status to “P360”. After reviewing the application, the caseworker marks it as either approved or unapproved in Excel. Depending on the verdict, a PDF approval or rejection letter is created. Norwegian applicants receive this letter via Digipost. For international applicants, the response is sent to the email address provided in the application. To document that the letter is sent, the caseworker once again updates the Excel spreadsheet. Finally, if deferment is given the caseworker opens FS and sets *møtt status* to R (deferment of one year) or T (deferment of two years). The BPMN model in *figure 4* illustrates the steps described in this subsection.

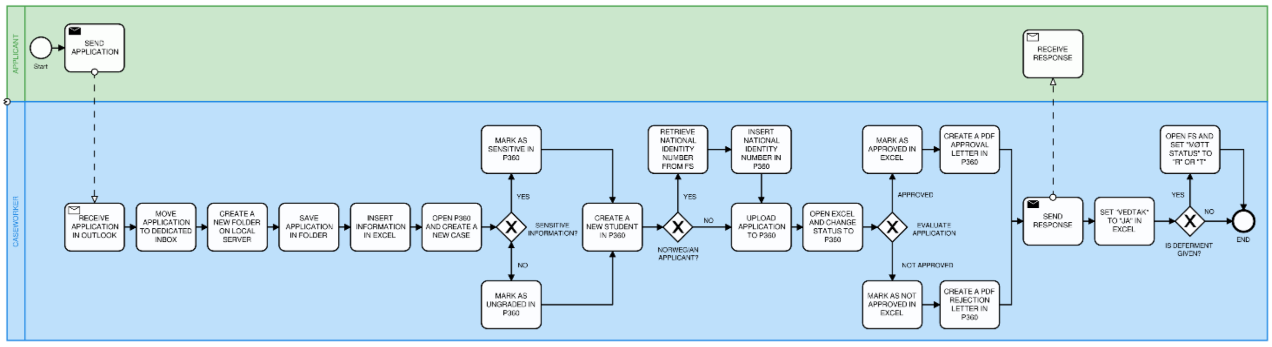

APPLICANT
Start
SEND APPLICATION
RECEIVE RESPONSE
CASEWORKER
RECEIVE APPLICATION IN OUTLOOK
MOVE APPLICATION TO DEDICATED INBOX
CREATE A NEW FOLDER ON LOCAL SERVER
SAVE APPLICATION IN FOLDER
INSERT INFORMATION IN EXCEL
OPEN P360 AND CREATE A NEW CASE
SENSITIVE INFORMATION?
YES
NO
MARK AS SENSITIVE IN P360
MARK AS UNGRADED IN P360
CREATE A NEW STUDENT IN P360
NORWEGIAN APPLICANT?
YES
NO
RETRIEVE NATIONAL IDENTITY NUMBER FROM FS
INSERT NATIONAL IDENTITY NUMBER IN P360
UPLOAD APPLICATION TO P360
OPEN EXCEL AND CHANGE STATUS TO P360
EVALUATE APPLICATION
APPROVED
NOT APPROVED
MARK AS APPROVED IN EXCEL
CREATE A PDF APPROVAL LETTER IN P360
MARK AS NOT APPROVED IN EXCEL
CREATE A PDF REJECTION LETTER IN P360
SEND RESPONSE
SET "VEDTAK" TO "JA" IN EXCEL
IS DEFERMENT GIVEN?
YES
NO
OPEN FS AND SET "MØTT STATUS" TO "R" OR "T"
END

Figure 4: BPMN of today’s process

5.2 Suggested Design of Artefact

Figure 5 displays the suggested design of the artefact. That is how we suggest the process of *deferment of study offer* would look like after the implementation of a low-code solution. The main goal of this design is to reduce the processing time by reducing the number of steps necessary to process an application. This is done by letting the caseworker work in only one system, thus creating a better overview and eliminating what we consider to be unnecessary steps. The BPMN model in *figure 5* displays the process from an applicant applies through the web form to deferment is eventually given in FS. Steps that are automated are marked with a script or a send/receive sign in the top left corner. The steps that are marked in red were not completed by the end of the development period. This is a result of us not being able to obtain the API keys required to integrate with P360 and FS. To access most API's, including the ones for FS and P360, one needs to insert an API key. An API key is a unique code that is passed into the application. The key is used to identify the user, developer or program calling, thus preventing malicious use of the API (RapidAPI, 2019). We will elaborate further on the potential for this integration in subsection *6.4 Further Development*.

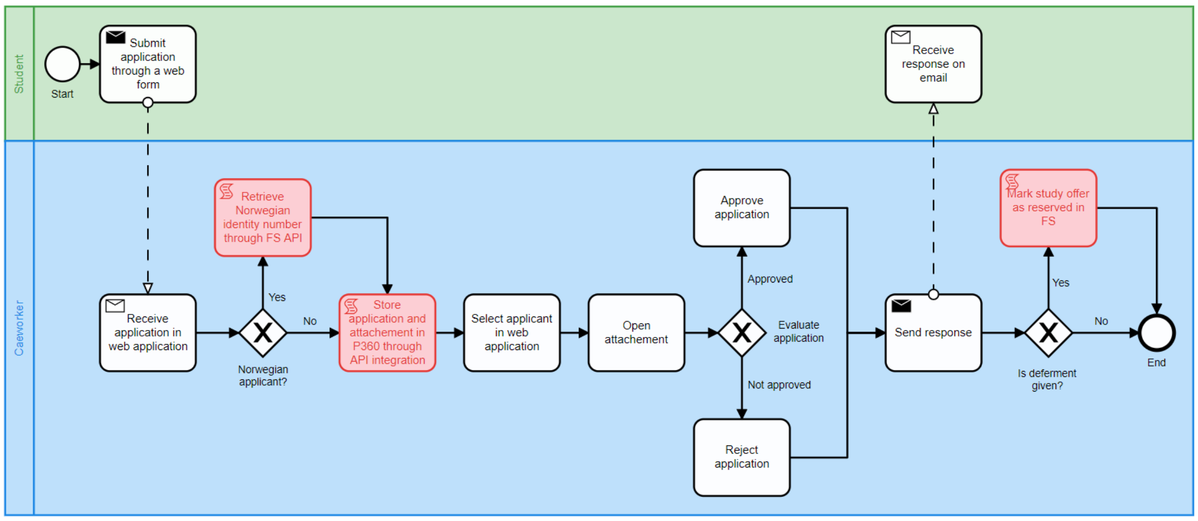

Student
Start
Submit application through a web form
Receive response on email
Caseworker
Receive application in web application
Retrieve Norwegian identity number through FS API
Yes
No
Norwegian applicant?
Store application and attachement in P360 through API integration
Select applicant in web application
Open attachement
Approve application
Approved
Evaluate application
Not approved
Reject application
Send response
Mark study offer as reserved in FS
Yes
No
Is deferment given?
End

Figure 5: BPMN for *deferment of study offer* with low-code

In this suggestion, the application is no longer received in Outlook, but rather in a new case management system. By having a complete overview of the applications in the case management system, we have removed the need for an Excel spreadsheet. Based on the input from the web form, the system automatically checks whether the applicant is a Norwegian citizen. If the applicant is Norwegian, the system uses the application number to retrieve the applicant’s national identity number through the FS API. By automatically retrieving the national identity number, we avoid asking the applicant to input this information in the web form. Even though a national identity number is not considered as sensitive information, the Admissions Office have clearly stated that they prefer not to receive this through a web form. Next, the system automatically registers a new person and case in P360. The attachment provided in the application is then saved to this case. In theory, one could build a low-code solution that is not depended on integration with external systems such as P360. However, we suggest the application and attachment are automatically stored in P360. This is in line with the recommendation from the Ministry of Education and Research, which states that P360 is one of two recommended archive systems (Kunnskapsdepartementet, 2017a).

The next step (*Select applicant in web application*) is the first where the caseworker actively handles the application. In this step, the caseworker opens the case management system and selects an applicant. Here the caseworker has a complete overview of the information provided in the application. The caseworker will then open the attachment and evaluate the content. Depending on the assessment, the caseworker will either press an acceptance or rejection button which automatically sends an email response to the applicant. The verdict, as well as the sending of the response, is automatically logged in the case management system, thus giving a clear overview of the applications that have been processed. If deferment is given the system will automatically mark the study offer as deferred in FS.

6. Development

In this chapter, we will discuss the development of our artefact. First, we will introduce the low-code development platform OutSystems before giving a more detailed explanation of the development process. We will then demonstrate our low-code application before discussing the potential for further development. It is important to state that the data displayed in this chapter is not based on real applications.

6.1 Development Platform

To develop our application, we have used the low-code development platform OutSystems. In this section, we introduce OutSystems and try to provide the reader with a basic understanding of the platform's capabilities and underlying logic.

6.1.1 Introduction to OutSystems

With OutSystems, applications are built and changed in a visual environment where the user can define the application's data model, business logic, workflow processes and user interfaces for both web and mobile devices. All development is done in a drag-and-drop development environment, which can be extended with custom code. Both Forrester and Gartner recognise OutSystems as one of the leading low-code development platforms (OutSystems, 2019c). We believe that developing on a recognised platform, such as OutSystems, provide a more realistic foundation for answering our research question. Before writing this thesis, we had no experience working with application development. To learn to develop in OutSystems, we attended several online courses as well as a seminar organised by Avo Consulting in Bergen.

6.1.2 Service Studio

The OutSystems development environment, *Service Studio,* is organised in four layers, *Processes, Interface, Logic* and *Data*. These layers provide the user with easy access to the elements that exist inside of Service Studio.

Processes

The process layer gives the developer information about logic and tasks that occur on an aggregate level. Inside processes, we have two major groups, processes and timers. Processes include business processes as well as human and automated tasks, which in return contains decisions, events and wait times. The second type of element is the timer, which is a scheduled action that can occur at a specific time and be rescheduled to occur daily, weekly or at other types of intervals. These actions can be attributed priorities so that if multiple actions must run at the same time, some can be attributed higher importance. They can also be attributed timeouts, stopping actions from running indefinitely.

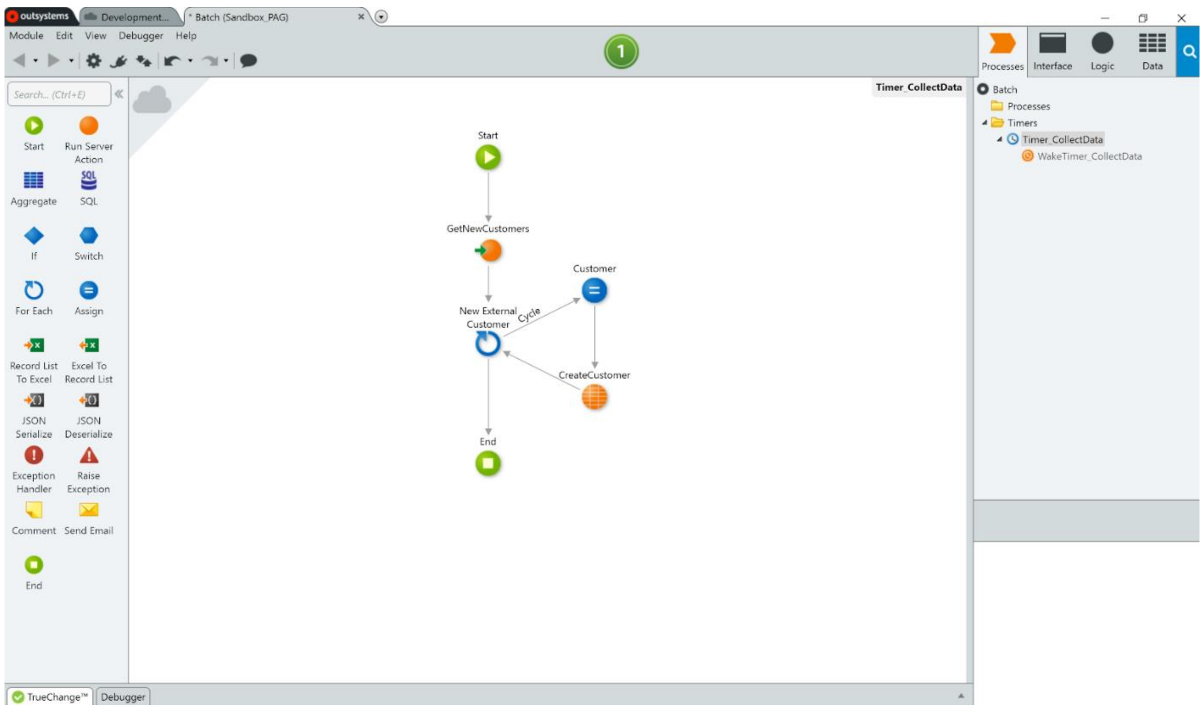


Figure 6: The process layer in Service Studio

Interface

The second layer is the interface layer. The interface focuses on the different components that make up the user interface. This includes user interface flows which are groups of individual screens. Besides giving an overview of the different screens, the interface layer includes images, and themes, which control the look and feel of the application. The layer also includes scripts. This may be any JavaScript resources that is be required for any of the elements in the user interface.

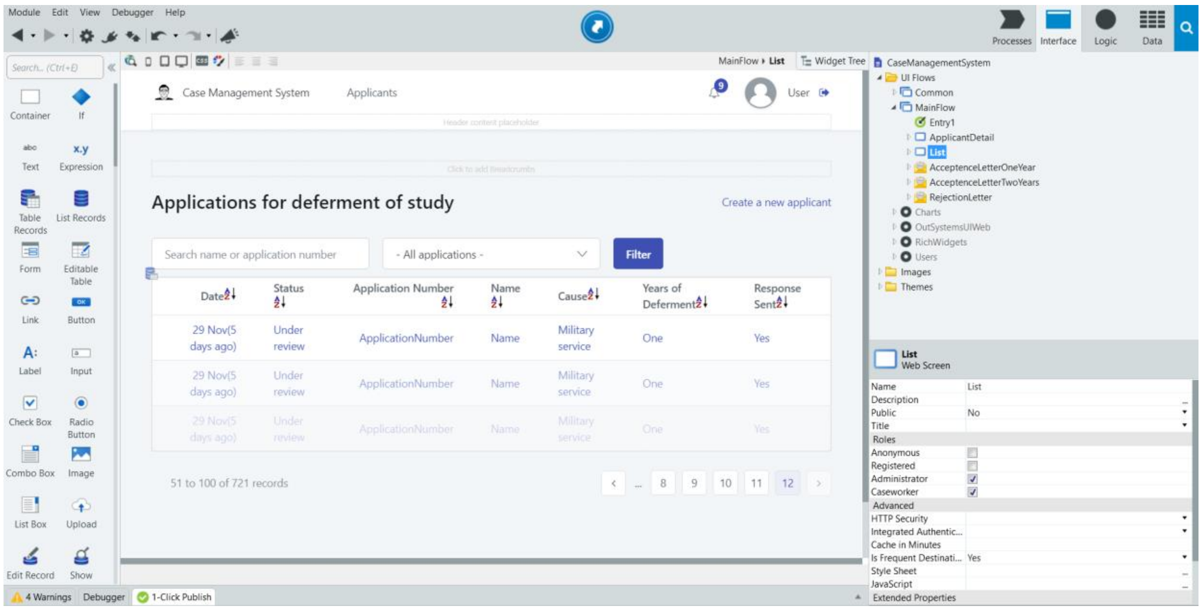


Figure 7: The interface layer in Service Studio

Logic

In the third layer, we have the logic layer which displayes the individual logic of each action. As we can se by *figure 8*, the logic is displayed by the use of a flowchart. The logic layer also provides easy access to external systems, including web services and enterprise systems like SAP. Because we have logic in all these sections, we also want to secure and limit who has access. This can be done by assigning different roles to different users. Also, if any exceptions need to be handled, we can easily add a set of exceptions in the logic layer.

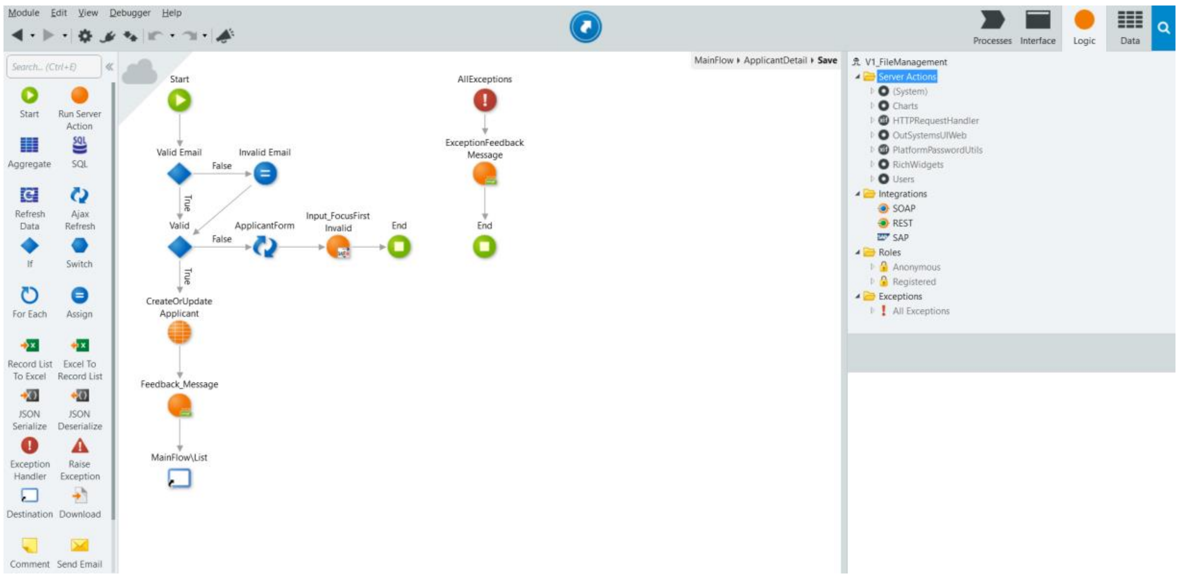
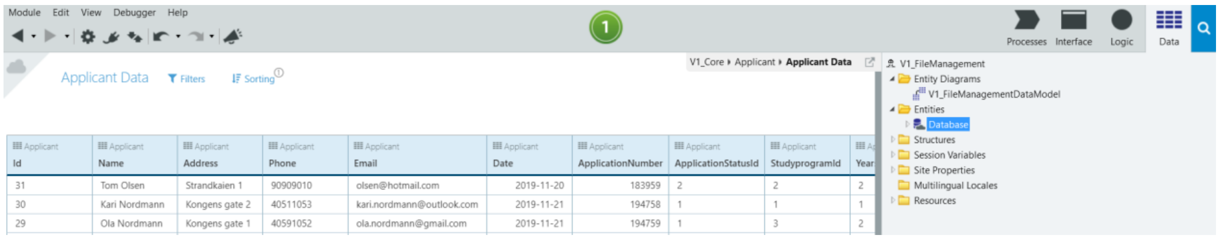

Module Edit View Debugger Help
Processes Interface Logic Data
Start Run Server Action Aggregate SQL Refresh Data Ajax Refresh If Switch For Each Assign Record List To Excel Excel To Record List JSON Serialize JSON Deserialize Exception Handler Raise Exception Destination Download Comment Send Email
MainFlow ▸ ApplicantDetail ▸ Save
Start
Valid Email
Invalid Email
False
True
Valid
ApplicantForm
Input_FocusFirst Invalid
End
CreateOrUpdate Applicant
Feedback_Message
MainFlow\List
AllExceptions
ExceptionFeedback Message
End
V1_FileManagement
Server Actions
(System)
Charts
HTTPRequestHandler
OutSystemsUIWeb
PlatformPasswordUtils
RichWidgets
Users
Integrations
SOAP
REST
SAP
Roles
Anonymous
Registered
Exceptions
All Exceptions

Figure 8: The logic layer in Service Studio

Data

Finally, we have the data layer. Here we can define the different entities that are available in the database. OutSystems define entities as “elements that allow you to persist information in the database and to implement your database model” (OutSystems, n.d.-a). If we wish to represent these visually, entity diagrams can be created. Besides entities that will be stored on a server, there are also structures which are in-memory representations of data. There are site properties, which are cross-application data, and then there are resources, which is any other type of data that do not fall into the other categories.



Applicant Id	Applicant Name	Applicant Address	Applicant Phone	Applicant Email	Applicant Date	Applicant ApplicationNumber	Applicant ApplicationStatusId	Applicant StudyprogramId	Year
31	Tom Olsen	Strandkaien 1	90909010	olsen@hotmail.com	2019-11-20	183959	2	2	2
30	Kari Nordmann	Kongens gate 2	40511053	kari.nordmann@outlook.com	2019-11-21	194758	1	1	1
29	Ola Nordmann	Kongens gate 1	40591052	ola.nordmann@gmail.com	2019-11-21	194759	1	3	2

Figure 9: The data layer in Service Studio

6.2 Development Process

In this subsection, we will describe the development process and the decisions which shaped our low-code application. We have structured our development in two sprints that both lasted one week. Before we started the development of our application, we had a walkthrough of the current process with a representative from the Admissions Office. Here we mapped the entire process, which resulted in the BPMN process shown in subsection *5.1 Current Process for Deferment of Study*. After we had mapped the process, we met with two consultants from Avo Consulting. Together we created an overview of the elements that needed to be included in our application, also known as the product backlog. Our solution is built in the free version of OutSystems. This implies that the data is saved in our personal environment on the OutSystems cloud. If the solution is to be implemented, the data would likely be stored on a local server (on-premise), in the cloud or on a combination of the two.

6.2.1 First Sprint

The first sprint lasted one week, from 4th to the 10th of November. The focus was on creating the main components of our solution. We created three separate components: a core, a web form and a case management system. The reasoning behind the separation of the components is primarily due to security risks. Neither the applicant nor the caseworker should be able to have direct access to the database itself but rather update the database from remote applications. The web form and case management system are separated to ensure that applicants do not have access to the submitted applications. *Figure 10* illustrates how the three components interact with each other: The web form updates the core, and then that data is displayed in the case management system. After the submitted applications have been reviewed in the case management system, the data is once again saved to the core component.

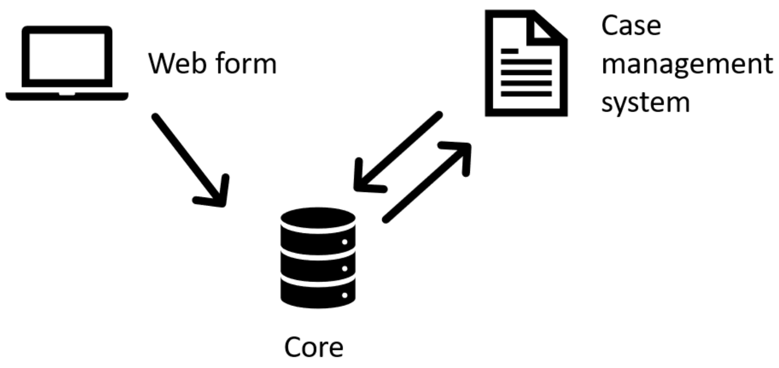


Figure 10: Illustration of the three components

Core

In the core, we define the entities that make up the foundation of our application. In our solution, we have included the entities *Applicant* and *ApplicantFile*. In the *Applicant* entity, we store the applicant's personal information, such as name, address, phone, email, and study program, as well as application-related information like application number, date of submission, application status, cause of deferment and if a response has been sent. In the *Applicant* entity, we have also added a comment attribute as well as an attribute to insert additional information in the email. These attributes allow the caseworker to leave a comment in both the case management system and in the email response. We believe these attributes can be useful if the caseworker would like to give a more extended explanation of the reasoning behind the assessment.

In the *ApplicantFile* entity, we store the documents that the applicants attach to their applications. To make sure that the attachment is connected to the correct applicant, the primary key in the *Applicant* entity (*ApplicantId*) is set as a foreign key in the *ApplicantFile* entity. This relationship can be viewed on the left-hand side of *figure 11*.

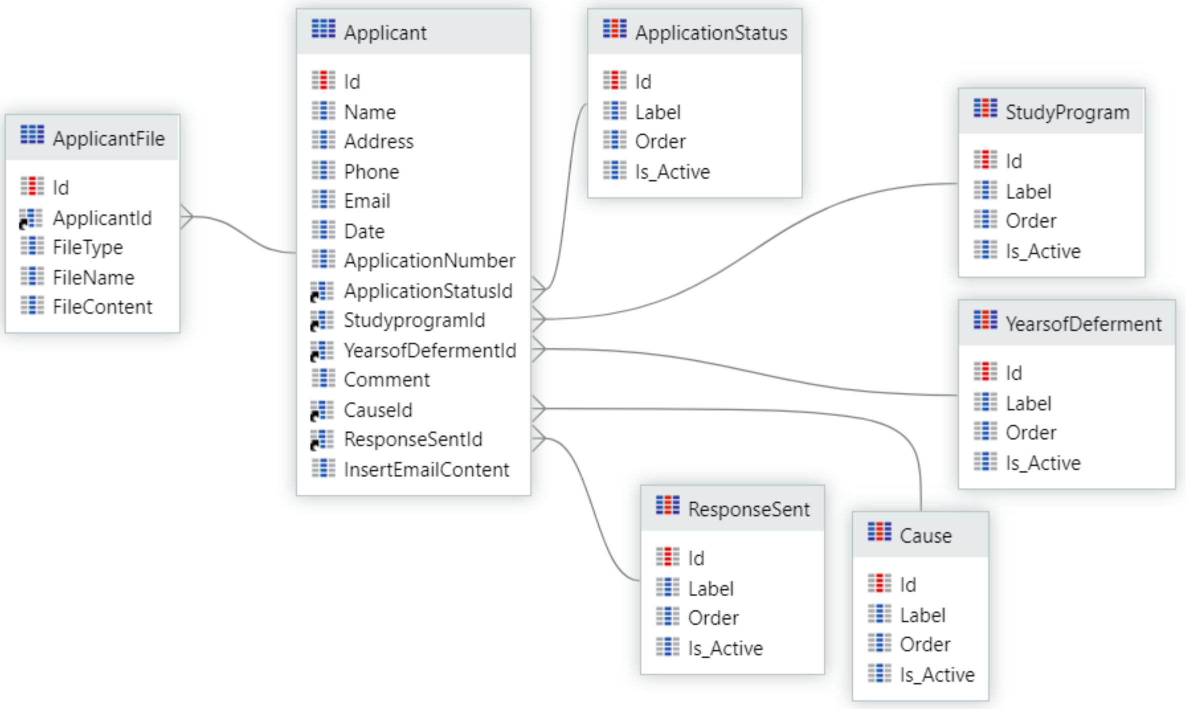


Figure 11: Entity Diagram

We have also created five static entities: *ApplicationStatus*, *StudyProgram*, *YearsofDeferment*, *Cause* and *ResponseSent*. Static entities are used when you need a predefined and constant set of values (OutSystems, n.d.-d). As displayed in *figure 11*, all static entities are connected to the *Applicant* entity through a foreign key.

The *ApplicationStatus* entity gives the caseworker an overview of which applications that have been processed, and which applications that are under review. It contains three records: *Under review*, *Approved* and *Declined*.

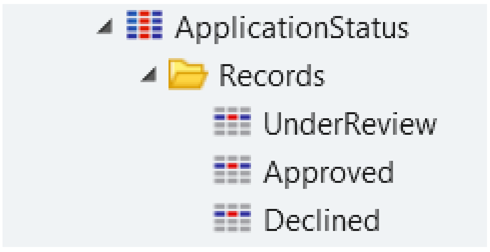


Figure 12: Application Status entity.

The *StudyProgram* entity holds the different study offers at NHH. This is Bachelor, Master's in Accounting (MRR), Master of Science for Norwegian students (MØA) and Master of Science for international students (MSc).

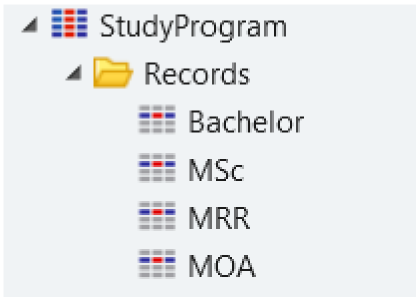


Figure 13: Study Program entity.

The *YearsofDeferment* entity refers to the number of years the applicant can withhold their study offer. Because it is only possible to receive deferment for one or two years, this entity only contains two records, One and Two.

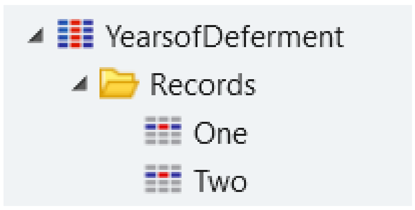
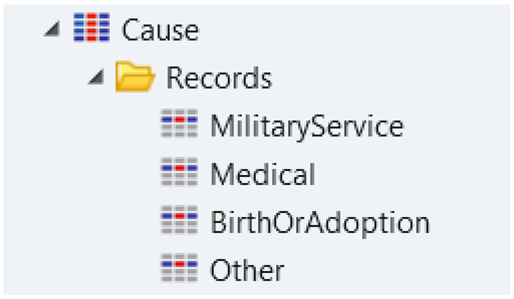


Figure 14: Years of deferment entity

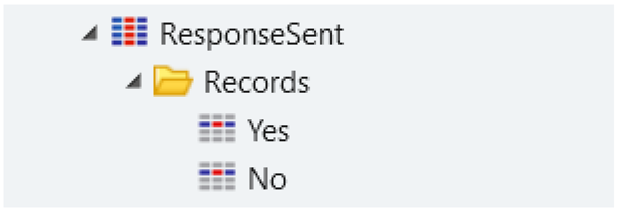
The *Cause* entity contains the cause of deferment. In conversation with representatives from the Admissions Office, it emerged that most students who apply for *deferment of study offer*, apply for three reasons, compulsory military service, medical issues and birth or adoption. We have therefore included four records in the *Cause* entity: Military Service, Medical and Birth or Adoption and Other. It is important to state that with OutSystems, one could easily add more records to an entity if deemed necessary. For example, adding more common causes for deferment.



Cause
Records
MilitaryService
Medical
BirthOrAdoption
Other

Figure 15: Cause of deferment entity

Finally, we have the *ResponeSent* entity. This entity tracks whether a response is sent to the applicant and contains two records: Yes and No.

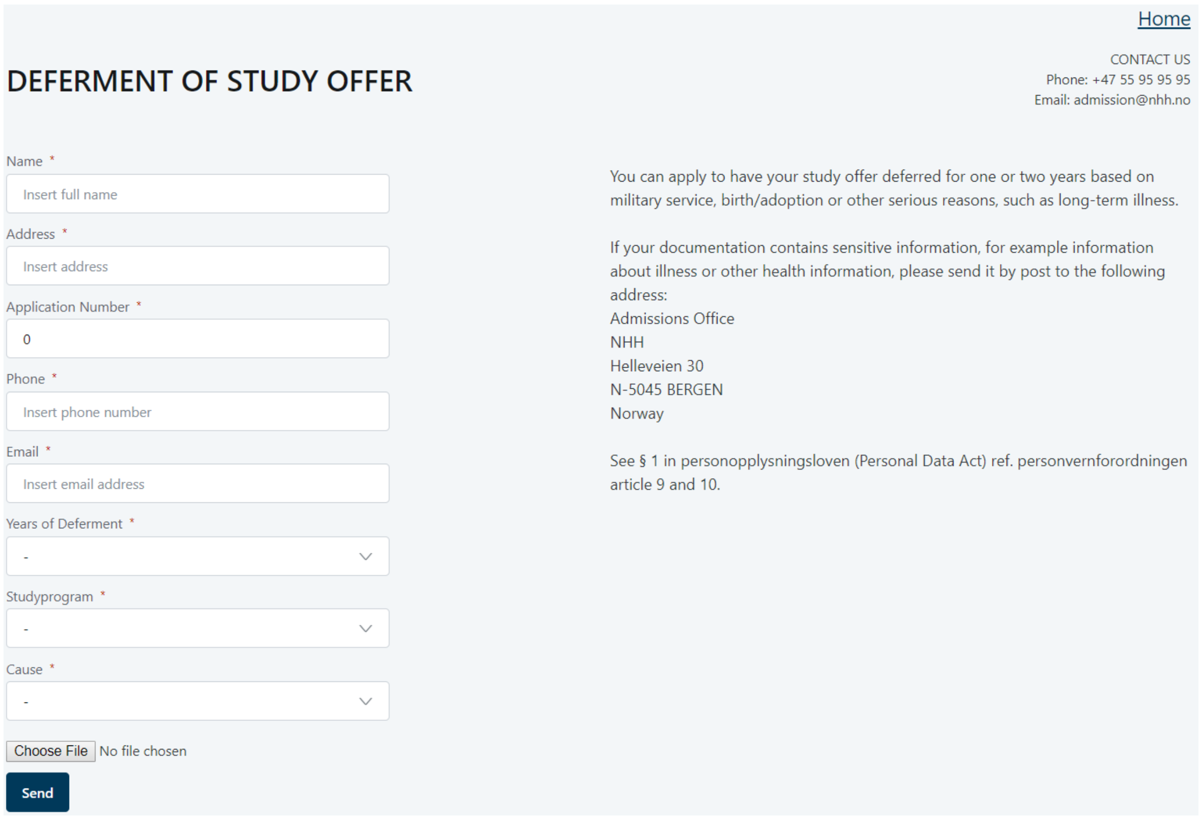


ResponseSent
Records
Yes
No

Figure 16: ResponseSent entity

Web form

Because OutSystems makes it simple to create a web form that automatically updates the database (in this case the core), we decided to replace the email-based web form that is in use today. Just like the web form that is currently in use, the new web form asks applicants to input their name, address, application number, phone number, email address, years of deferment, study program and cause of deferment. Applicants are also asked to upload a document that proves that they are entitled to receive deferment. If the document contains sensitive information, it shall not be uploaded, but rather sent by post. To avoid applicants sending sensitive information we have clearly stated what is considered sensitive and where they should send this information. Just like the web form that is in use by NHH today, our web form is available to the public. This means that anyone can access the website without having to log in with a username and password. The reasoning behind this decision is that many of the applicants are new students and therefore, do not have a student email or account registered. Making the applicants sign up for an account makes the process more time consuming for the applicants without gaining any real value for the Admissions Office. We also consider the likelihood of someone submitting a false application as small.



Home

DEFERMENT OF STUDY OFFER

CONTACT US
Phone: +47 55 95 95 95
Email: admission@nhh.no

Name *	Insert full name
Address *	Insert address
Application Number *	0
Phone *	Insert phone number
Email *	Insert email address
Years of Deferment *	-
Studyprogram *	-
Cause *	-

Choose File No file chosen

Send

You can apply to have your study offer deferred for one or two years based on military service, birth/adoption or other serious reasons, such as long-term illness.

If your documentation contains sensitive information, for example information about illness or other health information, please send it by post to the following address:
Admissions Office
NHH
Helleveien 30
N-5045 BERGEN
Norway

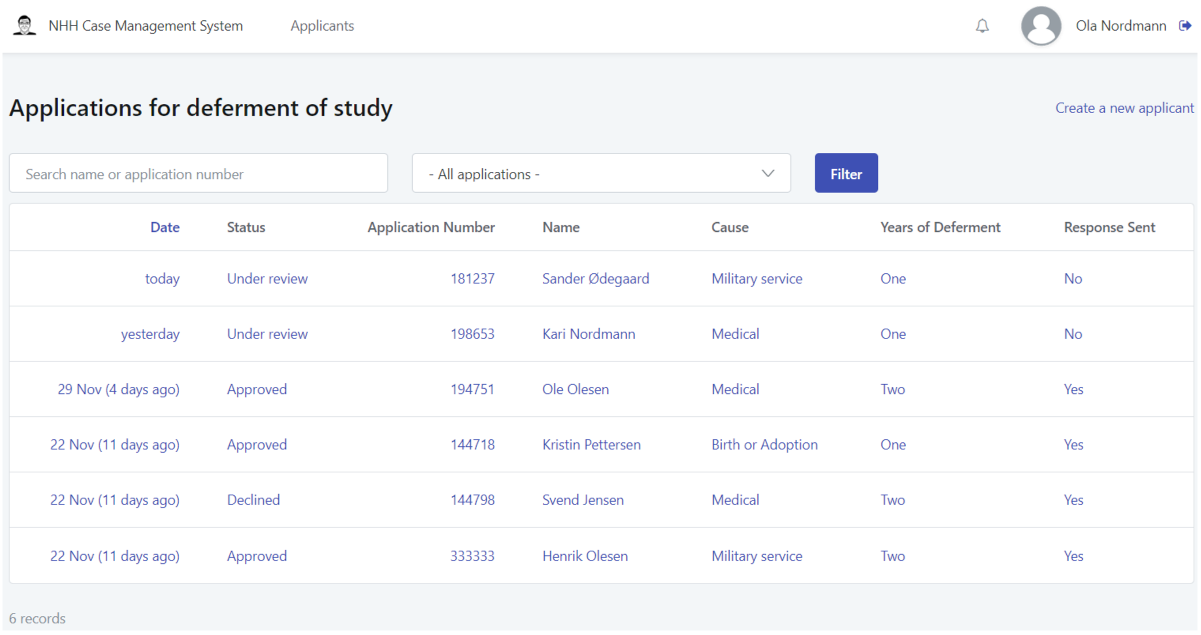
See § 1 in personopplysningsloven (Personal Data Act) ref. personvernforordningen article 9 and 10.

Figure 17: Web form

Case management system

When a student applies through the web form, the application is received in a case management system. Unlike the web form, the case management system is only available for users with a predefined role as a caseworker. This ensures that no outsiders have access to the submitted applications. The management of roles is done in the OutSystems Service Center. Service Center is a web console that enables the operational management of OutSystems applications. It provides the user with all logging and monitoring information related to the applications. The *application overview screen* in the case management system is viewed in *figure 18*. This screen includes a table with an overview of the submitted applications. As we can see in the figure below, are all applications automatically assigned to the date they were submitted. To enhance the user experience, we have added a search bar and a drop-down menu. The inclusion of a search bar lets the caseworker search for a specific applicant by both name and application number. This reduces the need to scroll through several pages of applications. The drop-down menu lets the caseworker filter between the application statuses. This feature makes it easy to see which applications have been processed and which are still under review.

In a meeting with the Admissions Office, it was stated that some applicants did not use the web form but instead contacted the Admissions Office directly by email. To allow the caseworker to easily archive applications that are not received through the web form, we have added a feature that allows the caseworker to add a new applicant directly in the case management system. The button to add a new applicant can be viewed in the right-hand side of *figure 18*.



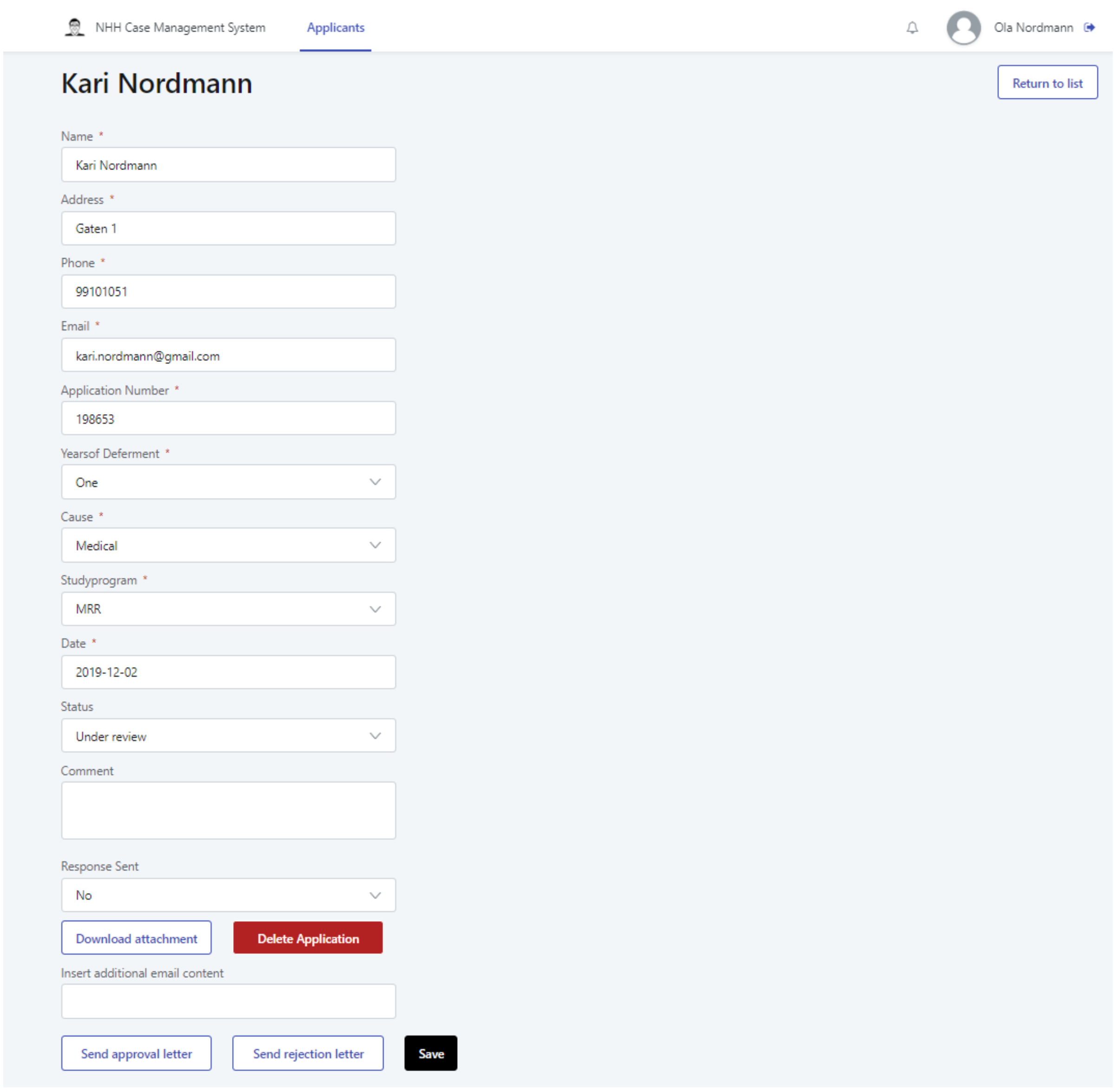
Applications for deferment of study

Date	Status	Application Number	Name	Cause	Years of Deferment	Response Sent
today	Under review	181237	Sander Ødegaard	Military service	One	No
yesterday	Under review	198653	Kari Nordmann	Medical	One	No
29 Nov (4 days ago)	Approved	194751	Ole Olesen	Medical	Two	Yes
22 Nov (11 days ago)	Approved	144718	Kristin Pettersen	Birth or Adoption	One	Yes
22 Nov (11 days ago)	Declined	144798	Svend Jensen	Medical	Two	Yes
22 Nov (11 days ago)	Approved	333333	Henrik Olesen	Military service	Two	Yes

6 records

Figure 18: Application Overview Screen

By pressing any of the rows on the *application overview screen*, the caseworker gets access to the *application detail screen.* As seen in *figure 18*, the *application detail screen* contains all the information that has been submitted by the applicant. As we consider it likely that some applications will contain typos and other errors, we have given the caseworker access to change any of the information provided in the application. When a new application is submitted, it is automatically assigned with the status “Under review”. In the application detail screen, the caseworker can change this status to either “Approved” or “Declined”. The caseworker can also leave a comment or input additional information to the email response.



Kari Nordmann

Name *	Kari Nordmann
Address *	Gaten 1
Phone *	99101051
Email *	kari.nordmann@gmail.com
Application Number *	198653
Yearsof Deferment *	One
Cause *	Medical
Studyprogram *	MRR
Date *	2019-12-02
Status	Under review
Comment	
Response Sent	No
Insert additional email content	

Figure 19: Application detail screen

As seen in *figure 19*, we have added a few buttons to the application detail screen. First, we have a return button in the upper right corner. By pressing this, the caseworker is sent back to the *application overview screen* without any changes being saved to the database. To download the documentation provided by the applicant, we have created a download attachment button. When pressed, this button triggers an action that searches the *ApplicantFile* entity for a file that is assigned with the same *ApplicantId* as the applicant the caseworker is currently assessing. The action then retrieves the file and downloads it to the caseworker's computer. The logic of the download attachment button is viewed in *figure 20*. The delete button, as the name suggests, deletes the application and its corresponding attachment from the database, while the save button saves any changes and returns the user to the *application* *overview screen*. Lastly, we have added a couple of buttons for sending a response to the applicant. By the end of the first sprint, the logic of these response buttons was not completed, and we, therefore, postponed this task to the second sprint.

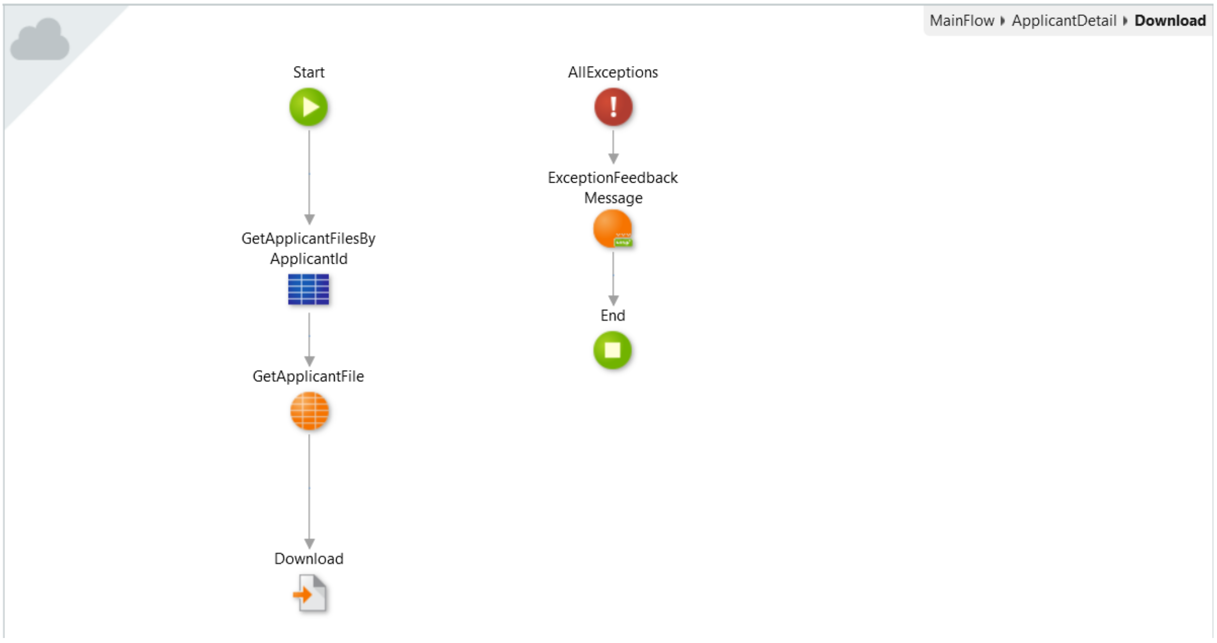
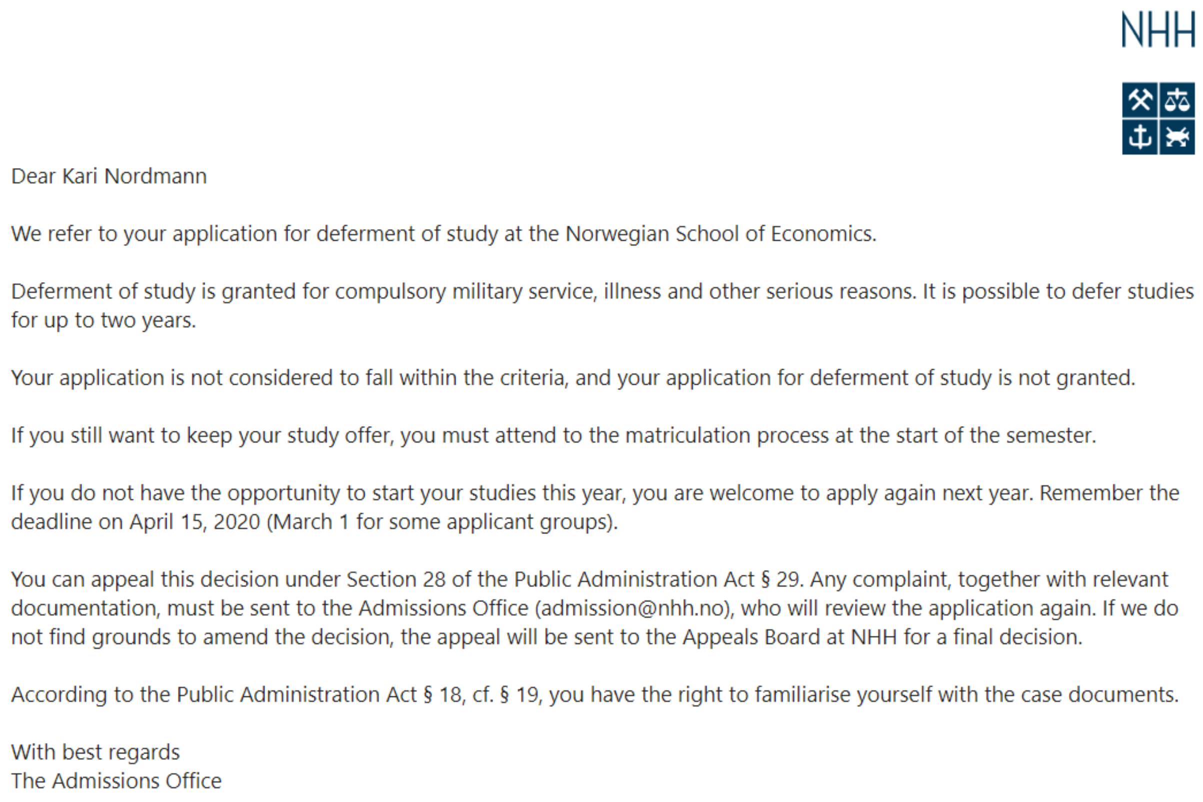

MainFlow ▸ ApplicantDetail ▸ Download
Start
GetApplicantFilesBy ApplicantId
GetApplicantFile
Download
AllExceptions
ExceptionFeedback Message
End

Figure 20: Logic of the download attachment button

6.2.2 Second Sprint

The second sprint lasted from the 18th to the 24th of November. We had initially planned to use this sprint to integrate our application with P360 and FS through an API. As mentioned in subsection *5.2 Suggested Design of Artefact,* we were not able to access the API keys required to integrate with P360 and FS. We will elaborate further on the potential for this integration in subsection *6.4 Further Development*. Instead of integrating with external systems, we used the second sprint to include elements that we were not able to include in the first sprint as well as making our solution more secure and user-friendly.

As mentioned in subsection *5.1 Current Process for Deferment of Study offer*, the Admissions Office is currently replying to Norwegian Students through the digital mailing service Digipost. As none of the responses contains sensitive information, we have, together with the Admissions Office, concluded that it is not necessary to use Digipost, making an email response a valid option. In the first sprint, we created two reply buttons, one for sending an approval letter, and one for sending a rejection letter. Just like web screens, OutSystems provides the developer with the opportunity to create and compose emails. Based on examples from the Admissions Office, we have composed a rejection letter and two approval letters. For the purpose of this thesis, all letters are written in English. However, if the solution is to be implemented, one could easily add a logic that would send responses written in Norwegian to the Norwegian applicants. To make the emails more personalised, they all contain an expression which retrieves the name of the applicant from the application form. This enables the emails to automatically include the applicant's name in the first sentence. The only difference in the two approval letters is the difference in length of deferent. The emails we have created are standardised and are therefore suitable in most situations. As mentioned earlier, the caseworker also has the opportunity to insert additional information in the email, for example, the reasoning behind the assessment. Still, in some cases, it will probably be necessary to compose a more customised response. The rejection letter and one of the approval letters can be viewed in *figure 21* and *22*. None of these examples includes a custom response.



NHH

Dear Kari Nordmann

We refer to your application for deferment of study at the Norwegian School of Economics.

Deferment of study is granted for compulsory military service, illness and other serious reasons. It is possible to defer studies for up to two years.

Your application is not considered to fall within the criteria, and your application for deferment of study is not granted.

If you still want to keep your study offer, you must attend to the matriculation process at the start of the semester.

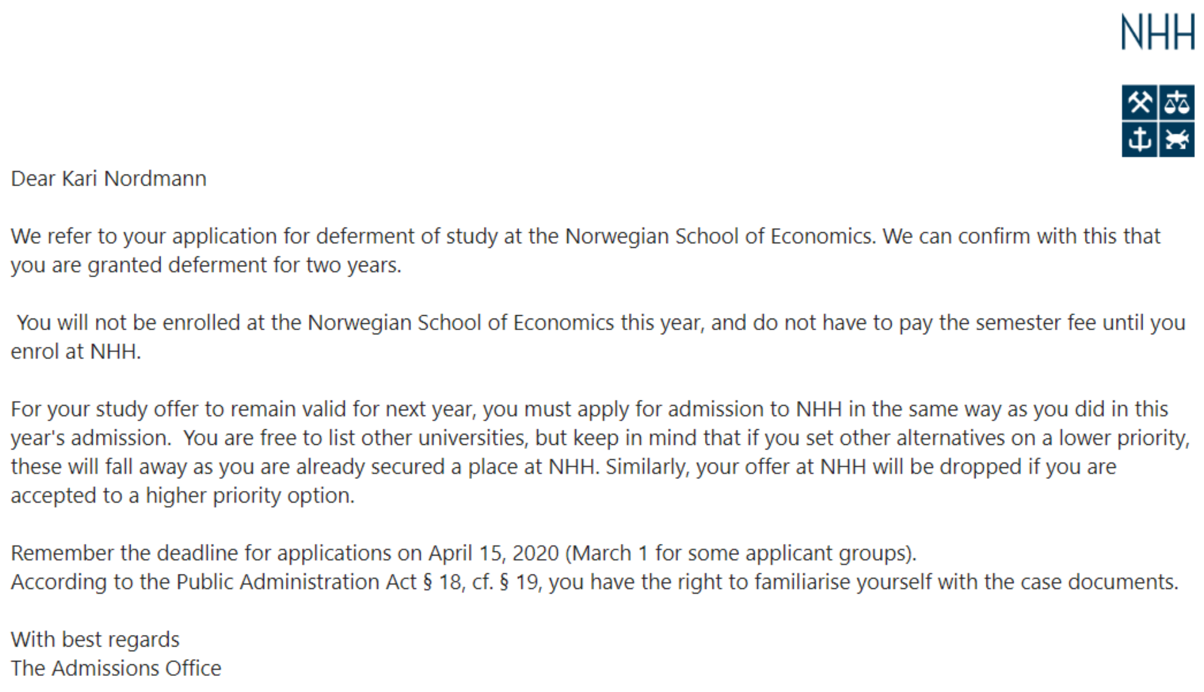
If you do not have the opportunity to start your studies this year, you are welcome to apply again next year. Remember the deadline on April 15, 2020 (March 1 for some applicant groups).

You can appeal this decision under Section 28 of the Public Administration Act § 29. Any complaint, together with relevant documentation, must be sent to the Admissions Office (admission@nhh.no), who will review the application again. If we do not find grounds to amend the decision, the appeal will be sent to the Appeals Board at NHH for a final decision.

According to the Public Administration Act § 18, cf. § 19, you have the right to familiarise yourself with the case documents.

With best regards
The Admissions Office

Figure 21: Rejection letter


NHH

Dear Kari Nordmann

We refer to your application for deferment of study at the Norwegian School of Economics. We can confirm with this that you are granted deferment for two years.

You will not be enrolled at the Norwegian School of Economics this year, and do not have to pay the semester fee until you enrol at NHH.

For your study offer to remain valid for next year, you must apply for admission to NHH in the same way as you did in this year's admission. You are free to list other universities, but keep in mind that if you set other alternatives on a lower priority, these will fall away as you are already secured a place at NHH. Similarly, your offer at NHH will be dropped if you are accepted to a higher priority option.

Remember the deadline for applications on April 15, 2020 (March 1 for some applicant groups).
According to the Public Administration Act § 18, cf. § 19, you have the right to familiarise yourself with the case documents.

With best regards
The Admissions Office

Figure 22: Approval letter

To be able to send email through our application, we needed to connect our OutSystems account to an SMTP server. This is done in the OutSystems Service Center. SMTP stands for Simple Mail Transfer Protocol and is the standard communication protocol used for sending and receiving email (Van Vleck, 2001). The SMTP server tells where the email should be sent, the authorisation information needed for the transaction, and the type of security needed (Tschabitscher, 2019). Since the Admissions Office is currently using Outlook, we have connected to the Outlook SMTP server. Within a minute of the caseworker pressing either of the two reply buttons, the applicant receives an email response to their application.

As we can see in *figure* 23 and *24*, the logic of the two reply buttons is almost the same. The most significant difference is the inclusion of an if statement in the send approval letter button. This if statement checks whether the applicant is given deferment for one or two years and then sends an email with the correct length of deferment. Inside the logic of the two buttons, we have added two server actions. Depending on which button that is pressed, the system automatically changes the application status to either Approved or Declined. In addition, the system sets the *ResponseSent* entity to Yes. Finally, the data on the applicant detail screen is refreshed and a feedback message confirms that the letter is sent. If the caseworker does not use either of the reply buttons, for example, if a custom response is necessary, the caseworker can manually mark the response as sent in the *applicant detail screen.*

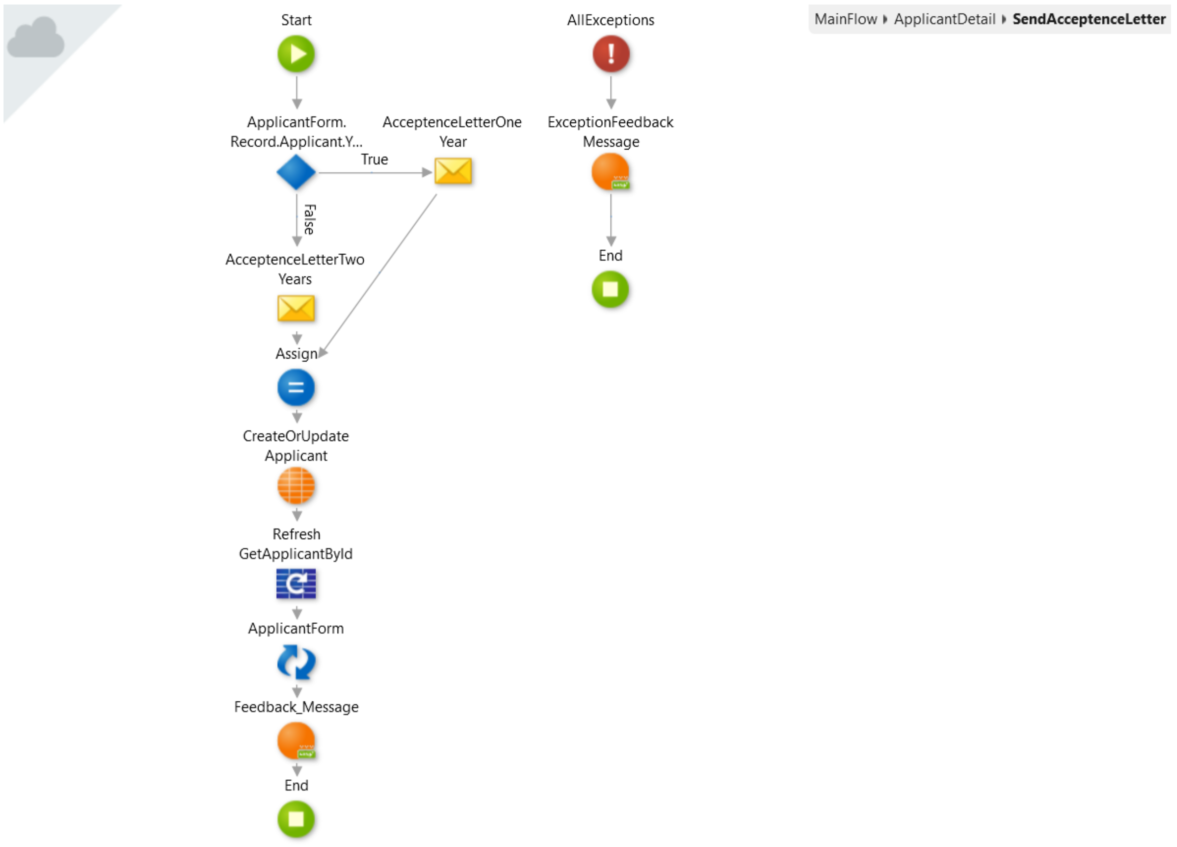


Figure 23: Logic of the send approval letter button

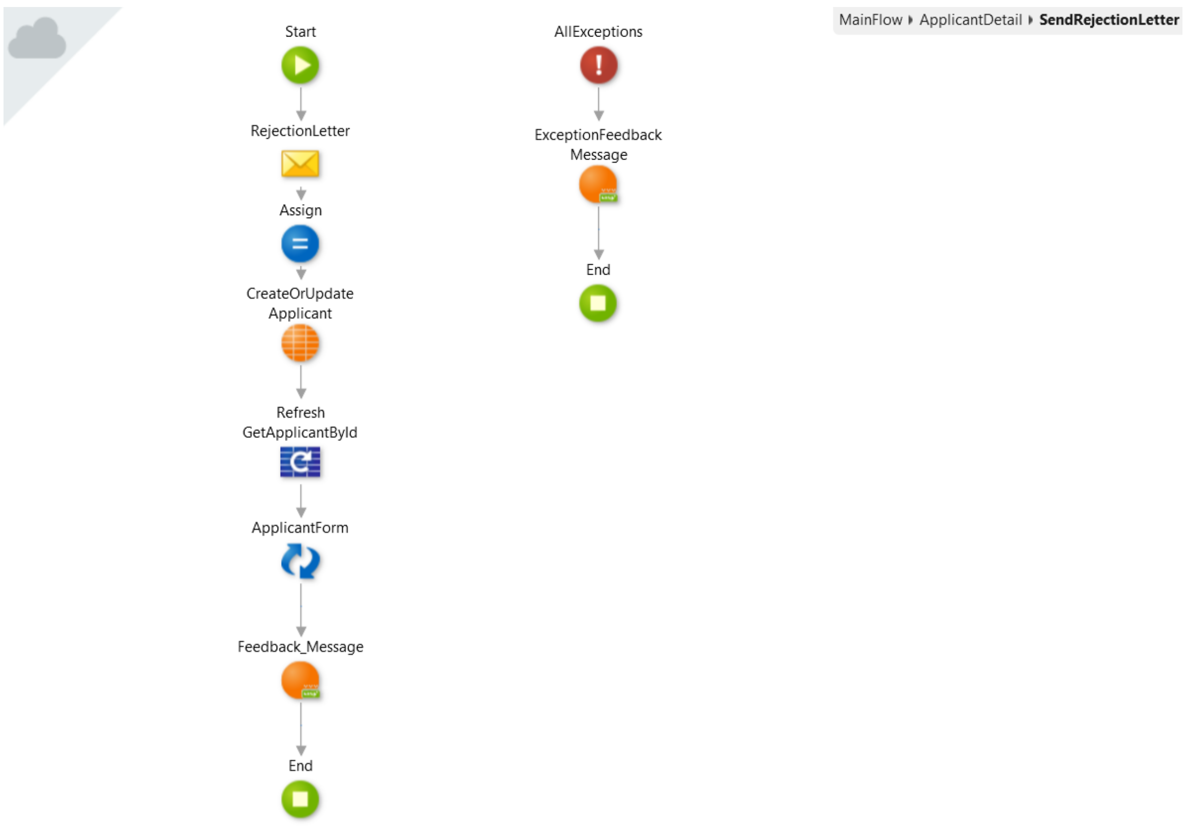
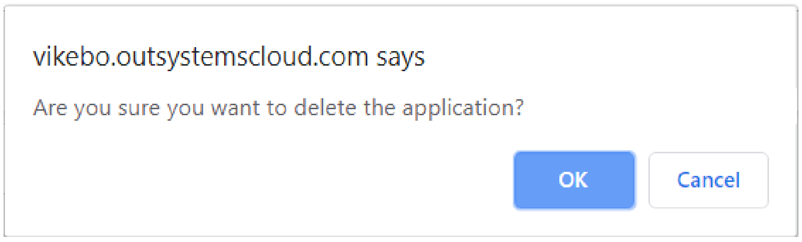


Figure 24: Logic of the send rejection letter button.

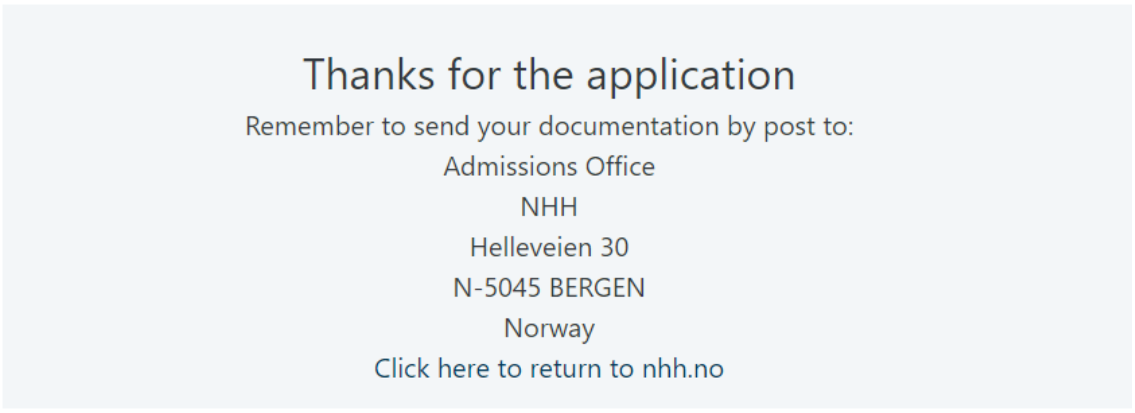
We have also added a confirmation message to three of the buttons. When the caseworker press either of the reply buttons or the delete application button, it will be displayed a popup message asking the casework to confirm that the input is correct. This confirmation message act as an extra layer of security, stopping the caseworker from accidentally deleting an application or sending the wrong response. The confirmation message for the delete application button can be viewed in *figure 25*.


vikebo.outsystemscloud.com says

Are you sure you want to delete the application?

Figure 25: Confirmation message for the delete application button

We did not only add new features to the case management system. In the second sprint, we also added two new screens to the web form. The first new screen is displayed to applicants who apply for deferment because of birth, adoption or medical issues. Because health information is regarded as sensitive, it shall not be uploaded but rather sent by post. A problem with the web form that is in use today is that it allows all applicants to upload documents, regardless of the information is sensitive or not. To prevent applicants from uploading sensitive information, we have included a statement that that checks which cause the applicant has chosen for deferment. If the applicant has chosen medical or birth or adoption as the cause for deferment, the web form will not upload the document but instead transfer the applicant to a new screen that asks the applicant to send the documentation by post. This screen can be viewed in *figure 26.*


Thanks for the application

Remember to send your documentation by post to:

Admissions Office

NHH

Helleveien 30

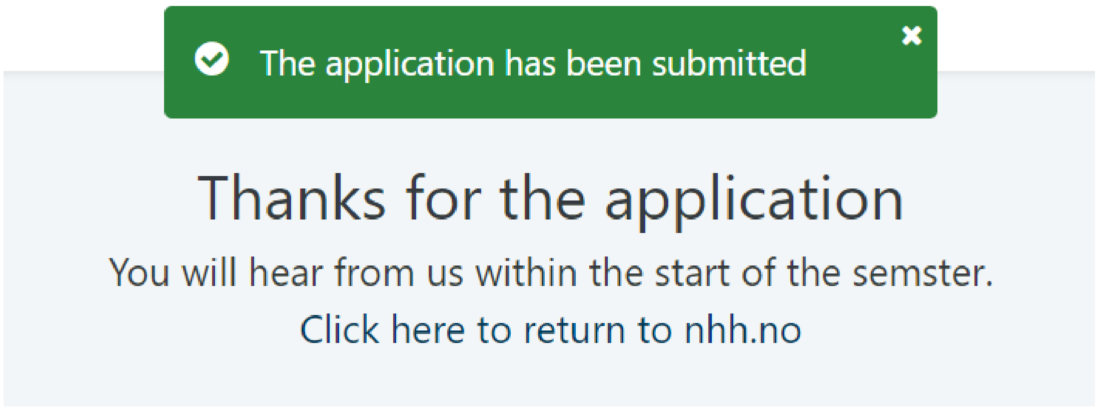
N-5045 BERGEN

Norway

Click here to return to nhh.no

Figure 26: Final screen after for students with medical causes, including birth and adoption

We have also added a new screen for applicants who do not need to send sensitive information. After the applicant has applied, he or she is transferred to a final screen that confirms that the application has been correctly submitted. From here, the applicant can press a link that redirects to NHH's front page. This final screen is displayed in *figure 27*.



The application has been submitted

Thanks for the application

You will hear from us within the start of the semster.

Click here to return to nhh.no

Figure 27: Final screen for applicants with non-medical causes

To enhance the security of our low-code solution, we have added HTTPS security to all web screens. HTTPS is a communication protocol that supports secure communication over the Internet. HTTPS uses HTTP in combination with Transport Layer Security (TLS) or Secure Sockets Layer (SSL), which encrypts the communication session so that unauthorised persons cannot intercept or change data during the transmission over open networks (Bartnes, 2019).

After the first sprint, we recognised that the web form would allow all file types to be uploaded through the web form. This created a significant security risk because anyone with bad intentions could potentially upload malware (malicious software) through the web form. To deal with this security loophole we have created a logic that checks whether the file is PDF document. If the file is not a PDF the applicant would get an error message, stating that the document must be a PDF. In this logic, we also check whether the applicant has forgotten to upload a document. If an applicant with a non-medical cause does not select a file to be uploaded, the same error message will be displayed. The complete logic behind the web form can be viewed in *figure 28*.

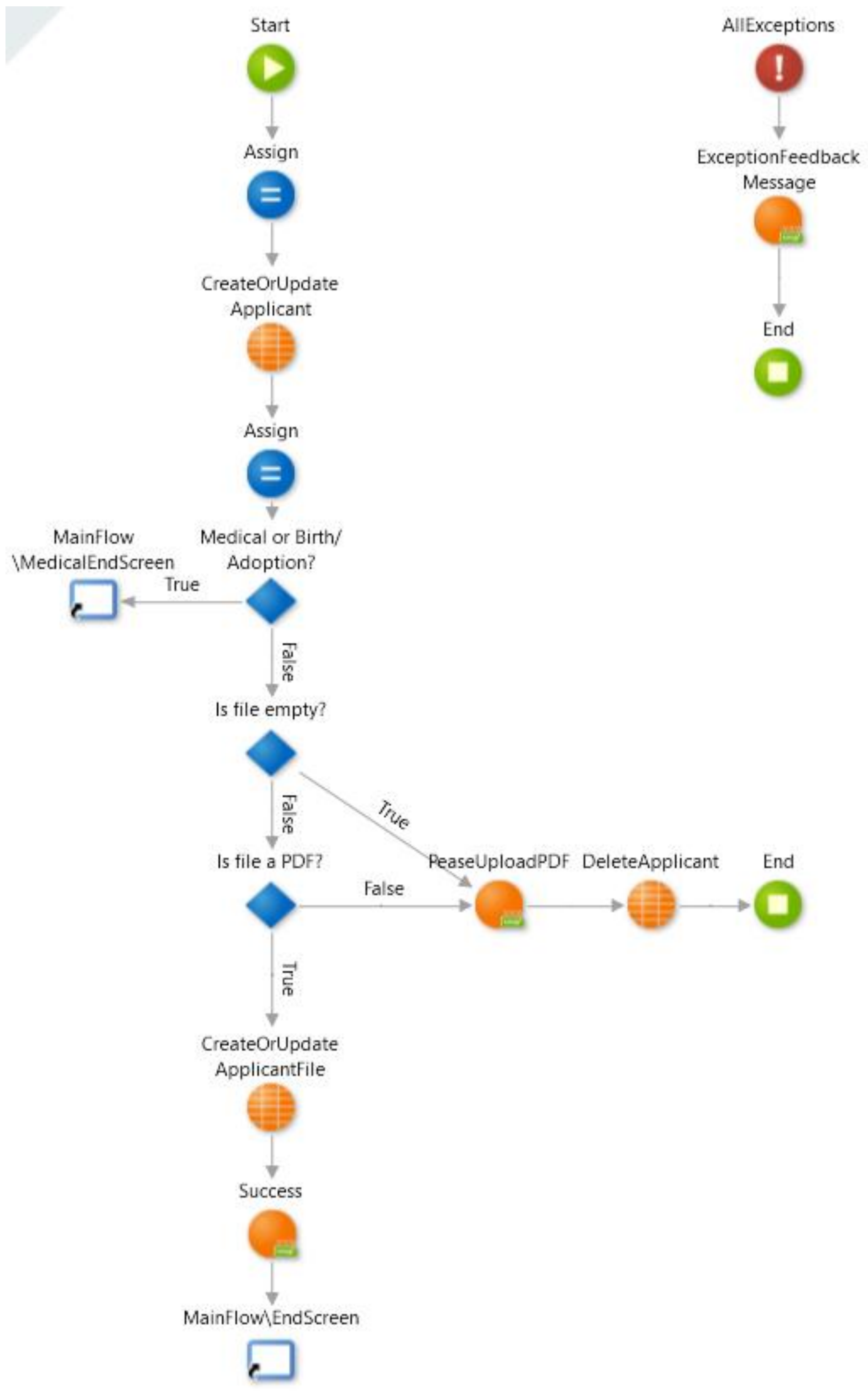


Figure 28: Logic of the web form

6.3 Finished Prototype

After the second sprint, we met with representatives from the Admissions office. Together we had a walkthrough of our low-code solution, demonstrating its capabilities and shortcomings. We will elaborate further on the feedback from the Admissions Office in chapter *7. Evaluation of the Artefact.* A video demonstrating our application in action can be viewed by following this link: https://vimeo.com/378607980. It is important to state that the video contains fictional data.

6.4 Further Development

The biggest shortcoming of our solution is the lack of integration with FS and P360. As mentioned earlier we have not been able to access the API keys required to connect with these APIs. To get the API keys that we required, the IT department at NHH contacted the vendor of P360. Although waiting several weeks, our request was left pending, even though the vendor was contacted several times by the IT department. Although FS has an API, we could not use it because it did not have a trial version, like P360. Also if we were given access to FS, we would be exposed to students personal information which could be in discord with Norwegian privacy laws.

In preparation for the second sprint, we read the API documentation for both P360 and FS. P360 was recently updated to a cloud service, resulting in a change in its API, while the FS API was launched as late as 2018. As of writing, the P360 API is based on the Simple Object Access Protocol (SOAP), while, the FS API is built on the REST architecture. The difference between these API standards are out of scope for this thesis, but when asked about whether OutSystems is compatible with these APIs, a leading low-code expert at Avo Consulting answered:

"*Yes, OutSystems can connect with virtually anything, and certainly via REST and SOAP Webservices*".

Because we never got started on the API integration, it is difficult to answer precisely how complex and time consuming the integration process would be. However, when asked about the difficulty of API integration on the OutSystems platform, another low-code expert at Avo Consulting answered that:

“*It is not unreasonable to assume that if you had access to the APIs, you would be able to integrate with them.”*

The Admissions Office have clearly stated that they prefer a solution that does not contain temporary storage. This is in line with the strategy report for digitalisation published by the Ministry of Education and Research (Kunnskapsdepartementet, 2017a), which states that information should only be stored and distributed from one location. As mentioned in subsection *5.2 Suggested Design of Artefact*, we suggest that the data is stored directly in P360, as it is one of two archive system that is recommended by the Ministry of Education and Research. When asked whether it would be possible to avoid temporary storage and instead save directly to P360, the same representative from Avo Consulting answered that:

“It is possible to avoid temporary storage. You must have some sort of a local variable, but you do not need to store it in a database. Creating your own OutSystems database is therefore not necessary”.

To further enhance our knowledge of API integration, we participated in several online courses on how to connect OutSystems with REST and SOAP APIs. Based on our knowledge as well as the feedback from the two low-code experts at Avo Consulting, we are confident that with access to the API keys we would be able to create a solution that is in line with the design we proposed in subsection *5.2 Suggested Design of Artefact.* As a result, the process of *deferment of study offer* would have been highly automated.

7. Evaluation of the Artefact

Artefacts in a design science research program should be evaluated both during and after development (Vaishnavi et al., 2019). While developing the artefact, we gained a more in-depth understanding of the inner environment as well as the outer environment. This led us to continually revisit, and evaluate, earlier stages of the DSR framework. In this chapter, we will only focus on the evaluation of our finished prototype. Since we did not receive the API keys required to develop the application in the way it was proposed, we were not able to perform any practical test that would have proven in a definitive manner that our application could have replaced the solution that is currently implemented at the Admissions Office. We have, however, performed several interviews to perform a qualitative evaluation of the artefact. The evaluation is based on the subsection *5.1.1 DSR Process Model*, as well as subsection *5.2 Interviews*. The informants in the evaluation were a couple of employees at the Admissions Office and two employees of Avo Consulting. In the evaluation meetings, we started by explaining to the participants how we understood the current process of *deferment of study offer,* and what kind of solution we believed would make the process more efficient. We then showed the participants the same prototype which was shown in the video link shared in subsection *6.3 Finished Prototype.*

7.1 General Utility, Quality and Efficiency

The informants who participated in the evaluation were in general positive about the proposed solution, but there were also ideas for improvements, as well as some concerns related to the *artefact*. One of the features that were requested by the manager of the Admissions Office was the possibility to delegate applications among the caseworkers. She also asked if it would be possible to create an interface to audit the caseworkers, one which would make it possible to see who assessed which applicant.

One of the informants at the Admissions Office made a comparison between the RPA solution that was developed last year, and the low-code solution that we developed. She said that in a way, the low-code application solved some of the challenges they experience with the current process. Having to copy and paste the information from the application into different systems is manual and time-consuming. Low-code technology enabled the removal of these steps by integrating the web-form with a case management system. The RPA solution, however, was able to automate the tasks that needed to be done in P360, something our prototype has still not proved that it can do. She said:

"I feel sorry for you that you were not able to gain access to P360 API. I feel that these two solutions (RPA and the low-code solution) *complement each other. The students who developed the RPA solution managed to automate a lot of the things that needed to be done in P360, while you found a nice way to integrate the steps in the process before data is stored in P360, as well as the steps that come after the application is evaluated".*

Another employee at the Admissions Office agreed:

"It is like you have solved part A and C, while they solved the part B. The best would be if one solution could solve them both"

At Avo Consulting, informants were positive and thought the suggested design could have been implemented. One concern that we brought up was whether it was possible to send information directly from a webform to P360 without having to store it locally first. For us, the application needed to have this functionality because this is considered best practice in the digitalisation strategy report from the Ministry of Education and Research (Kunnskapsdepartementet, 2017a). When asked about the ability to store data directly in P360, a consultant at Avo Consulting answered:

"It is not unreasonable to think that this would work out fine. If you have access to the API, then you will be able to do it. The data needs to be temporarily stored in the application when it is managed, but there is no need for a separate database".

Although we could not measure how efficient our *artefact* was, we did ask the Admissions Office how long it takes to evaluate an application for deferment. Since the evaluation time is different depending on what grounds the applicant applies for *deferment of study offer*, the office gave us an estimate of a student who applies based on obligatory military service. This is considered one of the least complicated applications to evaluate because the documentation that needs to be evaluated is very standardised. They estimated that one of these applications could be evaluated in 3 to 5 minutes. Considering all the steps that are present in the current process for *deferment of study offer,* we imagine that a similar application could have been evaluated in our web application in about a minute.

7.2 Privacy and Security Concerns

A topic that was thoroughly discussed at the Admissions Office was the feature of automatically generated acceptance- or rejection letters sent to the applicants. In the web application we created, the mail is sent through an outlook account. As mentioned earlier does Norwegian applicants receive a response in Digipost. Digipost offers a secure mailbox where the user needs double authentication with BankID or Buypass to gain access. The reason this service is used is that it is considered a more secure, formal, way of communicating. The user that sends the mail can be sure that the content is only sent to the intended person, while the receiver gets a high-security mailbox for important documents.

The participants liked the automatically generated letters, but they were not sure whether the response should be sent from an outlook account. Although the emails that are sent from the Admissions Office usually do not contain any information which is regarded as sensitive, the office was worried that applicants might think that the letter is sent from someone else than NHH. Sending mail through Digipost with our suggested solution would not have been a problem, but it cannot be done from the OutSystems platform alone. The current process sends Digipost though P360, which is made possible because Tieto has partnered with Posten Norge. To send the mail through the web application, we would have to do this through the P360 API, or the Digipost API.

A topic during the evaluation meeting with Avo Consulting was the IT security of our application. It was pointed out that, although we had taken some measures to make our web application secure, for example using a secure transfer protocol (HTTPS), there were still underlying concerns. As explained by one of the informants. IT security on the OutSystems platform can split into three aspects. The first concerns the protocol security, which we already had implemented. The second aspect relates to the fact that when the OutSystem platform is used in development, machine code is generates based on the actions the developer performs in the user interface. The user must trust that OutSystems generates a secure code. Although it is possible to see the code that is generated, it is not meant for humans to read, which makes it difficult to audit.

The third aspect is related to the way data is transferred between the three components of the web application, described at the beginning of subsection *6.2.1 First Sprint*. In our prototype, any caseworker who logs in to the *case management system* would have been able to read, delete or update any information stored in the database. This kind of access for every caseworker is unadvisable since it exposes the information system to the risk of being manipulated against what was intended. It is, therefore, best practice to give each employee in an organisation only the access that they need to perform their tasks while restricting them from doing anything they are not supposed to do (Lunsford & Collins, 2008). The employee at Avo Consulting suggested that we implemented a logical layer of CRUD operations. CRUD stands for *create*, *read*, *update*, *and delete*. These are types of operations that are allowed, for example, on a piece of data for a specific user. By implementing this, one can prepare for the worst by restricting the possible actions a user can perform and protect the data integrity in P360. When asked whether it would be more complex and time-consuming to develop an application with CRUD operations, an Avo-employee answered:

"I would say that it is time-saving to include CRUD operations. IT provides a better overview and is straightforward to build on".

8. Discussion

In this chapter, we will return to the research question presented in the introduction of the thesis:

"Are low-code development platforms a viable solution to contribute to digital transformation in the administration of higher educational institutions in Norway?"

Supplemented by our findings while developing the *artefact* in the DSR study, we will attempt to answer the research question above, and contribute to the research field of digital transformation. Digital transformation is a vast topic. Consequently, we will therefore not address all the aspects of digital transformation in the Higher Education Sector. Since low-code is a tool designed to develop new applications, we will focus on how low-code can contribute to digital transformation through application development. We will first discuss some of the challenge organisations experience regarding digital transformation. We will then discuss whether low-code technology has the potential to solve some of these challenges, whether it is economically viable, and if so, how low-code technology should be implemented in the sector of higher education in Norway.

8.1 Challenges of Digital Transformation

In a literature review that investigated 282 academic publications regarding digital transformation, there was identified two types of organisational barriers for digital transformation. *Inertia* is related to existing resources and capabilities in an organisation, which can create barriers for changes that might disrupt existing processes in the organisation (Vial, 2019). A known example is Kodak, who were not able to react to the disruptive technology of digital photography. Even though they owned many valuable patents that could have helped them get a head start on their competitors, Kodak was not able to change the organisational culture and decision making, which was built on the success of analogue photography (Lucas & Goh, 2009). The other type of barrier that was identified was *resistance.* This is a theme that relates to the resistance employees of an organisation can produce when disruptive technologies are introduced. This raises, for example, the issue with how, and how fast one should introduce new technology into an organisation (Vial, 2019).

Another way to investigate the challenges of digital transformation is to see what the organisations themselves experience as barriers to digital transformation. OutSystems conducts an annual survey to investigate, among other things, what organisations find the most challenging with digital transformation and application development. The survey is called *The State of Application Development*. In 2019 it was filled out by 3300 IT professionals spread across multiple industries and continents (OutSystems, 2019a). Although we consider this source to a bit controversial since it was financed by a company selling low-code platforms, we found the way the survey was conducted satisfying. To mitigate bias in their survey, OutSystems promoted it primarily to IT professionals who were not OutSystems customers. The numbers of the survey will also be affected by geography since 70% of respondents were located in either North America or Europe. The largest responding industries were *software,* 20%, *Technology/Computers/telecoms* and *Internet,* 16%, *Consultant/Consultancy/SI,* 13%, *Government and Education,* 10% (OutSystems, 2019a).

As a part of the survey, respondents were asked to identify the top three challenges that delayed the delivery of web and mobile applications (OutSystems, 2019a). We have decided to elaborate on the six first causes, displayed in *figure 29.*

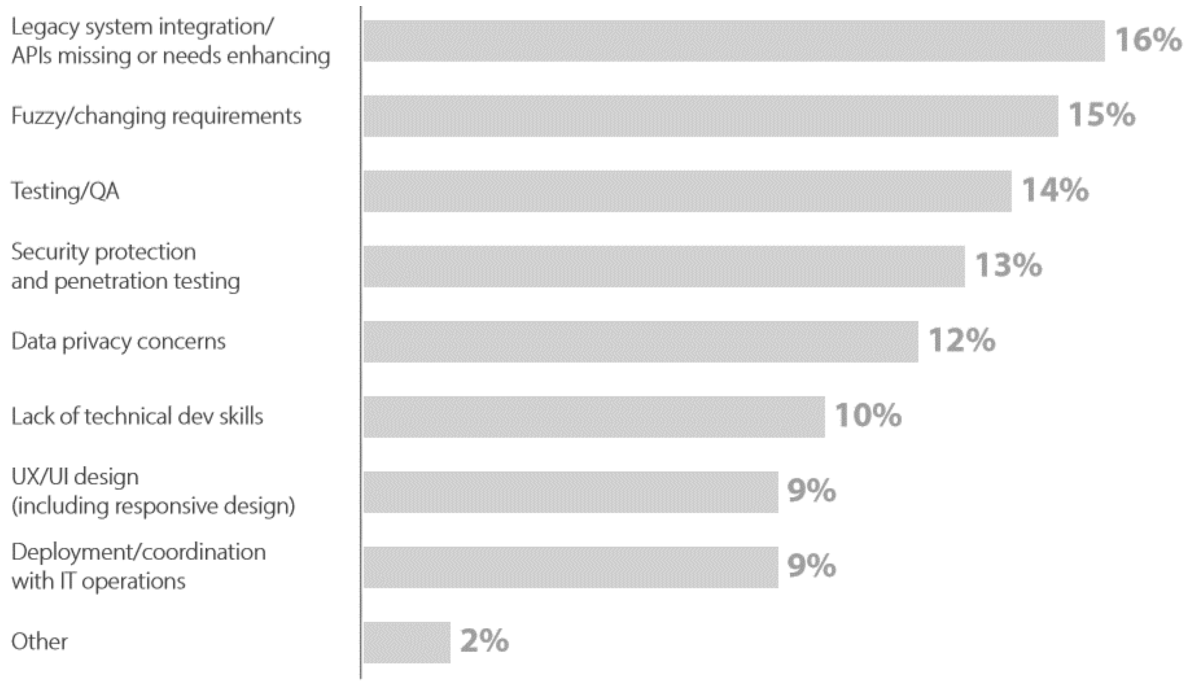


Figure 29 (OutSystems, 2019a) Top Causes of Application Delivery Delays

Although both P360 and FS had API integration during the development of our application, this was not the case until recently. For example, the API for *FS* was launched as late as 2018. *FS* was also a challenge for the students who developed the RPA solution for the process of *deferment of study offer* (Johnson & Eide, 2018). As well as not having access to the API for this system, it was also challenging to integrate with using an RPA solution. Their problem was not only an API problem but one that made any integration with *FS* difficult.

Even though a system has API solutions, it does not mean that a developer will gain access to this service. Our experience with not getting hold of the necessary API keys to P360 is an example of this. Although the vendor was contacted several times over a period of several weeks, the key was never received. During a presentation of our project for Avo Consulting, we mentioned the difficulty in receiving the API key. One of the consultants attending the presentation found this amusing, because our experience, was according to him, very similar to what developers experience in real business scenarios.

When developing applications, *changing requirements* are not unusual. Some of the most frequent reasons for changing requirements are that they are overlooked during planning, market shifts, miscommunication between stakeholders, changes within the organisation, or changes in laws and regulations (Bigelow, 2019). In our development, we experienced how easy it is to overlook details during planning. Although our *artefact* was not going to be put to actual use, we still got feedback during the evaluation regarding features that we were not aware was necessary for the application to perform in a satisfactory manner.

A lack of *software and quality assurance* can be costly both for an organisation. A report from 2002 by the Institute of Standards and Technology, estimated that software failures cost the US economy about 0.6 % of its yearly gross domestic product. The report also suggests that a third of these costs could be eliminated with testing infrastructure that enables earlier or more effective identification and removal of software defects (NIST, 2002). Developers using traditional code must design a set of tests, including test inputs and expected results, to remove bugs before the software release. Creating these tests for modern software is labour intensive and complex work. As a consequence, there are systems on the market for automatic testing of written code (Hossain, 2018).

Security Protection, Data Privacy are topics that are becoming increasingly important in application development. As described in *2.2 information systems,* IT architectures are becoming more integrated, as well as more connected to the internet. The more processes that are dependent on the internet, the more opportunities exist for hackers. This is because there are more potential points of entry to an organisations information system (Matthews, 2019). Data is also becoming more valuable. A natural consequence of this is that the stakes are higher regarding cybersecurity. The more an organisation depends on data, the more damaging an attack can be (Matthews, 2019). In 2016, Cyber Security Ventures predicted that cybercrime will cost the global economy 6 trillion dollars by 2021, up from 3 trillion in 2015 (Morgan, 2016).

It was also reported in the *2019 Cyberthreat Defense Report*, that 78% of organisations that were surveyed in the report believed that a cyberattack had already hit their organisation, and two-thirds believed it would happen again in 2019 (Cyberedge Group, 2019). When organisations were asked which security process they were most struggling with, they answered that app development and testing were causing the most headache, a response they also had given the two previous years (Cyberedge Group, 2019). In 2018 we got an example of how poor web and mobile application design can be a security risk for a company. Using only 22 lines of code, a group of hackers were able to steal personal information from 380.000 British Airways Customers. Amongst the information that was stolen, was names, addresses, and credit card details. Because the applications were not designed securely, hackers were able to add their own code to the applications existing code and change the webpage's behaviour. As a result, any data that was sent through the applications was compromised (Newman, 2018).

In the OutSystems survey, 10 % of the respondents named a lack of *technical dev skill* as the top cause of application delivery delays. Only 15% of the respondents described the recruitment of application developers as easy. The type of IT professionals that were most difficult to recruit were specialists in machine learning, cybersecurity, and internet of things (OutSystems, 2019a). If we consider the lack of human capital in general, then this was also a topic we identified at NHH. When asked about the challenges of digital transformation at NHH, a member of NHH's Digitalisation Committee writes:

"The biggest challenge is probably a lack of people who can lead and execute digitalisation projects and as well as the ability to make organisational changes that technology enables".

NHH is not alone in looking for this kind of knowledge. In Siemens annual survey about digitalisation in Norway, expertise in digital solutions was named the most significant barrier to digital transformation (Siemens, 2019).

8.2 Technological Viability

In the previous subsection, we examined some of the challenge organisations face in their digital transformation journey. In the context of our research question, we will explore the technological viability of low-code development platforms. This includes discussing whether low-code can solve some of the challenges described in the previous subsection.

8.2.1 Changing Requirements and Testing

Low-code platforms promise speedy delivery of small mobile- and web applications, as well as larger enterprise systems. Because of a growing focus on development delivery time, organisations have, in recent years, looked to low-code platforms to accelerate development (Richardson & Rymer, 2016a). A part of the reason why low-code development platforms grow in popularity is their modularity, as shown in chapter *8. Development*. Since low-code applications are made up of precoded modules, these modules do not have to be coded over again which might reduce the challenges mentioned regarding changing requirements. This emphasised by one of the consultants at Avo Consulting:

"The thing about low-code solutions is that so much of the things that people want to develop has already been done by someone else. Instead of writing the code over again, you can reuse code, and spend your time on something else".

The modularity and debugging features in low-code platforms have the potential of reducing the time and resources spent on debugging and testing. This is because every module in the platform is already coded efficiently and tested for defects (Jeffrey Martin, 2018). The platforms also have debugging features which makes it easier for the developer to know whether something will stop the application from working. For example, OutSystems quality assurance feature (OutSystems, n.d.-b). During the development of the low-code solution for NHH, we were given visual warnings at the bottom of the development screen every time an error occurred. If the error messages were selected, the platform would show us what was missing, and a suggested input to resolve the issue. Some argue, however, that the low-code platforms are not transparent enough, and that this makes debugging features like the one supplied by OuySystems, problematic. Although some low-code platforms can display the machine code after applications are published, the algorithm that produces the code, as well as the code behind debugging features are hidden from the developer. This is not an issue when the debugging feature discovers a known problem. However, when development hits a barrier that is not foreseen by the platform, the machine code makes it harder for the developer to solve the issue than if she had used traditional code (Wayner, 2019).

8.2.2 Security & Privacy Concerns

For low-code to be a viable option to contribute to digital transformation in the sector of higher education, it must comply with the information security requirements set by the Ministry of Education and Research. These requirements are based on national regulations and guidelines and must be understood as minimum requirements concerning information security. The ministry emphasises that the institution's awareness of information security plays an essential part in achieving the goals of digitalisation strategies and strategic initiatives (Kunnskapsdepartementet, 2017a).

As mentioned in subsection *8.1 Challenges of Digital Transformation,* application development and testing were named the security processes, in which organisations were struggling the most. These findings align with the findings from the report *The State of Application Development 2019.* In the report, security protection and penetration testing were named as one of the top challenges that slowed down application delivery. In addition, concerns about the security of the applications created were reported as one of the main reason's organisations were not adopting low-code platforms (OutSystems, 2019a).

One of the security concerns related to low-code development is that it might be difficult for organisations to get oversight on what their employees are building (Korolov, 2019). Part of this concern has to do with the problem of shadow IT. *Shadow IT* refers to "information technology applications and infrastructure that are managed and utilised without the knowledge of the enterprise's IT department" (Stroud, n.d.). Shadow IT is generally a result of business experts trying to fill gaps in the application portfolio (J. R. Rymer, 2018). The lack of oversight distinguishes shadow IT from citizen development.

Analyst at Gartner, Jason Wong states that lack of visibility is especially a concern for on-premise low-code solutions. *"If you install a rapid application development tool on a desktop* *and build apps, IT does not have any visibility"*. However moving to a cloud-based low-code solution can improve visibility, making it easier to apply governance and restrict access, thus increasing security (Korolov, 2019).

Another concern regarding low-code platforms is the security of sensitive information. Depending on the low-code platform, the organisation may choose to restrict access to some parts of the data. However, achieving adequate data segregation requires implementing access and role definitions, tasks that are generally outside the scope of the average citizen developer (Korolov, 2019). Independent low-code researcher Nigel Warren writes that it is not the responsibility of citizen developers to create secure applications. Instead, it is the responsibility of the IT department to procure and govern a platform that ensures its users only deploy secure applications (Warren, 2018).

Another security concern is that in some cases, the code and security controls that are provided by a low-code platform is not visible to the developer. This lack of transparency takes away some control from security teams. To learn how secure those platforms are, organisations may have to rely on third-party security audits, security and compliance certifications, service level agreements, and cybersecurity insurance (Korolov, 2019). However, some vendors try to make their code more transparent. For example, OutSystems generate .Net code, which allows customers to use third-party software to check whether the code is secure (OutSystems, n.d.-c).

That some low-code vendors do not provide this kind of transparency does not necessarily mean that security will suffer. In an interview, vice president of product strategy at IT-security firm Capsule8, Kelly Shortridge, says that she thinks that users, in general, are better off using a standard platform instead of writing the code themselves. She argues that if there is discovered any vulnerability in a component, the low-code vendors will realise a patch which would automatically update all applications that use that component (Korolov, 2019).

8.2.3 Complexity of Development

The State of Application Development 2019 reports that development skills are in short supply. Only 15% of respondents described the recruitment of application developers as easy, and for many specialities, recruitment was described as hard or very hard (OutSystems, 2019b)**.** Senior Director at the American software company Pegasystems, Sid Misra, writes that low-code can help bridge an organisation’s skilled developer shortage gap. Instead of relying merely on short-staffed professional programmers, low-code enables non-programmers to participate and contribute to application development (Misra, 2018). The CEO of low-code development platform Appian, Matt Calkins, has stated that they are “making it as simple as possible to build powerful software” (Barker, 2017). As two business students with no previous experience in application development, this thesis works as an interesting case for answering whether low-code is as easy to learn as some vendors suggest. Even though we had never developed an application before, both of us had some experience in data analytics and statistical programming. This experience might have contributed to a faster learning span, compared to a person with no programming experience.

Prior to project start-up, each of us spent approximately 50 hours on learning OutSystems. This includes both online courses as well as a seminar organised by Avo Consulting in Bergen. By being familiar with statistical programming, we know how time-consuming learning a programming language can be. It is hard to precisely estimate how long time it would take get the skills necessary to build a similar application with traditional code. However, we are confident that we would not be able to learn a traditional programming language as fast as we learned OutSystems. In particular, we would like to highlight the OutSystems Logic layer. By illustrating the logic with a flowchart instead of traditional code and syntax, the Logic layer makes it considerably easier for people without a technical background to understand and cooperate on the development.

At no stage during the development did we feel that OutSystems lacked the features necessary to complete our application. The main factors that prevented us from creating an application that was compliant with the suggested design were a lack of time and access to the API keys. Even though we had the option to add code by hand, we did not find it necessary for our purpose. However, as mentioned in subsection *2.3 Low-code Development Platforms,* there are more aspects to application development than just writing code. Even though we tried to follow best practice, there were moments where we felt we lacked the knowledge necessary to develop an application that met the standards necessary for implementation. For example, we found it difficult to know whether our application was compliant with the Admissions Office security requirements.

Our lack of experience in application development was evident in the evaluation meeting with the consultants from Avo Consulting. As mentioned in *subsection 7.2*, we did not include CRUD operations, which would have been necessary if the application were to be implemented to protect the integrity of the data. One of the consultants emphasised that even though OutSystems supports CRUD operations, it is the developer’s responsibility to implement it. The consultant believes that low-code has been portrayed as more accessible than it really is:

“You cannot forget the fundamental principles of programming and development, just because you have a low-code platform. Low-code is a good tool to help development, but you still need a good developer, and good design as well”.

We also asked the consultant on his thoughts on citizen development:

“If a businessperson wants to express her ideas, then she is more likely to be able to do this through low-code, than with traditional code. However, I do not think we will ever get to the point where everyone can code, because things also need to be done in a certain, correct, way”.

Even though the consultant at Avo does not believe that we will get to a point where everyone can code, we are confident that most people with a desire to participate in application development can learn the basics of low-code. This significantly increases the number of people who can assist in digital transformation.

8.2.4 Vendor Lock-in

In the report, *The State of Application Development 2019*, 37 % of respondents who were not using a low-code platform answered that concerns about vendor lock-in were one of the main barriers preventing them from adopting low-code. Vendor lock-in is a state where the customer is dependent on a vendor for products and services and is unable to use another vendor without substantial switching costs (Kratzke, 2014). We asked one of the consultants at Avo what his thoughts were on low-code and vendor lock-in:

“In a way, you get locked to the platform you choose. In OutSystems it is possible to retrieve the code, but as the code is written by a computer, it is somewhat difficult to interpret. However, if you have written everything in Java, you are in many ways locked into Java as well. This is a concern, no matter what software you choose”.

8.3 Economic Viability

As mentioned in subsection *2.3 Low-code Development Platforms*, the low-code market consists of over 100 vendors whom all provide platforms with different capabilities and pricing models. With so many alternatives, the customer can select platforms in all price ranges, stretching from free versions intended for small and medium-sized organisations, to enterprise plans that cost hundreds of thousands of Norwegian kroner (NOK) a month. However, the pricing models are often complex, making it difficult to compare vendors. John Rymer at Forrester Research writes that uncertainty about pricing is a barrier to adoption of low-code, and encourage vendors to make pricing and cost information fully transparent (J. Rymer & Seguin, 2019). In 2019 Rymer contributed to a report on the thirteen most significant low-code vendors. In this report, not a single vendor received top remarks on their pricing model (J. R. Rymer & Koplowitz, 2019). One of the problems contributing to this complexity is that some vendors charge fees depending on the number of registered end-users, while others charge depending on the complexity of the tool. OutSystems, for example, provides the customer with three options:

1) A free version where the customer can use the modelling environment, but not set the application into production. The free version also limits the registered end-user capacity at 100.

2) The Enterprise plan at approximately 60 000 NOK a month enables the customer with their own environment for testing and production, as well as a dedicated OutSystems Cloud to run applications. The registered end-user capacity starts at 100, with the option to add more.

3) The Universal plan starts at approximately 140 000 NOK a month and provides the same features as the Enterprise plan, but with unlimited registered end-user capacity.

Because the Admissions Office only processed 88 applications for deferment of study in 2019, it is probably not economically viable to implement a low-code solution for this process alone. In discussions with the Admissions Office, it was stated that they had several processes that they wanted to automate. If the scale of these processes is large enough to offset the cost of implementation, subscription and maintenance, a low-code solution could be a viable option.

However, it is important to state that other technologies also should be considered when deciding if low-code is the right option to contribute to digital transformation in the sector for higher education. The most natural comparison, at least in administrative work, is RPA, which has already proven its viability in automating processes in the administration of higher educational institutions in Norway (Johnson & Eide, 2018).

Avo Consulting, which is a supplier of both RPA and low-code solutions, writes in an email that an RPA licence is usually less expensive than that of a low-code development platform, such as OutSystems. However, depending on the size of the project, an organisation might need several RPA licenses, making an economic comparison case depended. They also emphasise that low-code and RPA are different products with different purpose and application.

8.4 Level of Implementation.

In their strategy report on digitalisation in institutions of higher educations, the Norwegian Ministry of Education and Research writes that support functions should be realised as joint services if it can be documented that this results in increased cost-effectiveness or better services. In our effort to answer whether low-code is a viable solution for institutions of higher education, an interesting question is whether low-code should be developed locally at every institution or in cooperation between the institutions.

As of writing, the Norwegian higher education sector consists of 10 universities, five public university colleges and nine specialised university institutions with state ownership. If each institution is to develop its own low-code solution, the sector will have used resources to develop 24 almost identical solutions. In addition, one has to address the expenses regarding the management of all these solutions. As a result, digitalisation in the sector might be more expensive than necessary.

Since the majority of higher education institutions in Norway are public, this should imply that most support functions are equal. Furthermore, as long as the tasks are equal, our opinion is that it would be more economical if institutions cooperate on low-code implementation. However, this assumes that information systems and work processes are standardised across institutions. In a paper outlying the challenges of reusing software robots in Norwegian municipalities, researchers Andreas Ulfsten and Jon Iden had three main findings (Ulfsten & Iden, 2018):

1) The municipalities used different case management and archive systems.

2) The same tasks were performed differently across municipalities.

3) The same information systems were configured differently across municipalities.

Even though this paper analyses robotisation in Norwegian municipalities, the authors write that the findings do not only apply to robotisation but can be extrapolated to digitalisation in general. We believe these findings might translate to institutions of higher education. However, further research is necessary to conclude if this is the case. If the findings do translate, it might be challenging to develop a low-code solution that fits the need of all institutions of higher education. To get an expert's view on this issue, we asked one of the low-code consultants at Avo Consulting whether NHH should initiate their own low-code implementation or if the development should be conducted in a collaboration between all institutions in the sector.

"Ideally, the development should be organised in a collaboration between several institutions. The downside to this is the local variations in how tasks are performed. It is hard for someone in, for example, Oslo to know the process in each institution. For the kind of process that you are developing, it should be possible to share the solution between the institutions. However, a general rule is that the differences between institutions in the sector are bigger than most people realise".

As mentioned in chapter *3. Digitalisation in the Norwegian Higher Educational Sector*, the Ministry of Education and Research encourage the higher education sector to digitalise and automate support functions in order to increase capacity in core activities. In subsection *8.1 Challenges of Digital Transformation,* we introduced *inertia* and *resistance* as two organisational barriers that affect digital transformation. If low-code is to be implemented to support administrative work, we might see both structural changes, as well as the removal of certain tasks. In this context, it is interesting to examine what NHH as an institution as well as employees think of adopting new digital tools to streamline administrative processes. A member of NHH's Digitalisation Committee writes in an email that there is a strong will on the part of both management and employees:

"Many people see the opportunities and want to contribute".

He also writes that systematic digitalisation is one of five strategic focus areas in NHH's strategy for 2018-2021. This includes, among other things, the establishment of a digitalisation committee as a step towards the overall management of NHH's investment in digitalisation. The committee will consist of staff who have solid digital expertise, as well as a good insight into the institution’s activities.

The manager at the Admissions Office was also positive regarding how the removal of manual processing steps could enable them to focus on other tasks:

«We will be able to spend more time on demanding tasks, increase communication with applicants, and spend time improving other processes. The increased capacity will also help us stay up to date on hearings and other academic changes”.

Today the universities in Bergen, Oslo, Tromsø and Trondheim have joined forces in what is known as the BOTT-cooperation. The purpose of the BOTT-cooperation is to strengthen the participating organisations' ability to provide administrative and technical services that support the organisations’ primary activities. The idea is that the implementation of the same information systems in institutions with similar needs enable the institutions to focus on primary activities (NTNU, n.d.). In the Ministry of Education and Research's Digitalisation Strategy for the higher education sector from 2017 to 2021, BOTT is given the responsibility to look at solutions that can be shared across the entire sector (Kunnskapsdepartementet, 2017a). By experiencing first-hand the benefits of low-code development, we suggest that low-code should be considered as a viable solution when the BOTT-cooperation evaluates different solutions that can be shared across higher educational institutions.

In their paper on robotisation in Norwegian municipalities, Ulfsten and Iden also introduced the possibility of a digital divide where small and medium-sized municipalities will not have the resources necessary for digitalisation (Ulfsten & Iden, 2018). We believe this discussion could be transferred to the sector of higher education. Without national cooperation between the institutions, we might see a digital divide where small and medium-sized institutions will be most affected. If smaller institutions cannot benefit from what large institutions do, they might rely on a less efficient administration. This can affect students by making the services worse because employees must spend a large part of their time on administrative tasks.

As mentioned in subsection *2.2. Information Systems*, Bygstad suggests that lightweight IT should be developed outside the IT department. However, unlike most lightweight IT, our experience is that low-code is more complex and therefore would be challenging to develop independently of the IT department. Regardless of whether low-code solutions are developed at a standalone institution or through cooperation between institutions, we recommend that the underlying IT-infrastructure, including database management and integration with external systems, is handled by IT-professionals. With oversight by the IT department, non-professionals will have the ability to build applications on top of the existing IT-infrastructure.

9. Conclusion

In this thesis, we have researched low-code development platforms' viability to contribute to digital transformation in the Norwegian sector of higher education. The background for our research is the 2017-2021 digitalisation strategy report from the Norwegian Ministry of Education and Research. The strategy report states that educational institutions are expected to take advantage of the opportunities provided by digitalisation to streamline administrative support functions and ensure proper management.

To answer our research question, we have used Design Science Research as our research methodology. The goal of DSR is to achieve knowledge and understanding of a problem domain by building an application of a designed artefact. In the context of our research, this has meant developing a low-code application for the process of *deferment of study offer* at the Norwegian School of Economics. To document the current process, as well as modelling a new suggested design for the process, we have used the Business Process Model and Notation Framework. After development, the artefact has been evaluated by industry experts to provide insight into the general utility, quality and efficiency, as well as the privacy and security concerns.

The artefact has demonstrated that it is possible, within a short amount of time, to develop a low-code solution that removes most manual processing steps, thus reducing processing time, while still maintaining a satisfactory overview. Our findings suggest that low-code development platforms can enable non-professional developers to contribute to application development, thus increasing the number of people that can assist in the digital transformation. However, we argue that most low-code development processes are still dependent on IT professionals running the underlying IT infrastructure. We suggest that shared low-code development between institutions of higher education can reduce cost and development time, but differences in existing information systems and work processes might reduce the possibility of shared development among institutions.

Through the development of our artefact, we have found that low-code development platforms are a viable solution to contribute to digital transformation in the administration of higher educational institutions in Norway.

Due to time constraints, we were not able to go into detail of all topics that were uncovered during our research. As of writing, there is little research regarding the viability of low-code solutions in general. Further research on the viability of low-code in both the higher education sector and other sectors would be interesting. In particular, we are curious to know how low-code could be applied in other parts of the higher education sector, such as research and education. There is also little research on the long-term benefits of low-code implementation, and we, therefore, suggest further research into benefits realisation management of low-code solutions.

10. Bibliography

Barker, C. (2017). Low code development: Is this the future of enterprise apps? Retrieved from ZDNet website: https://www.zdnet.com/article/low-code-development-is-this-the-future-of-enterprise-apps/

Bartnes, M. (2019). HTTPS. In *Store Norske Leksikon* (p. 1). Retrieved from https://snl.no/HTTPS

Bigelow, S. J. (2019). How to tame ever-changing requirements in software development. Retrieved from TechTarget website: https://searchsoftwarequality.techtarget.com/tip/How-to-tame-ever-changing-requirements-in-software-development

Bostrom, R. P., & Heinen, J. S. (1977). MIS Problems and Failures: A Socio-Technical Perspective, Part II: The Application of Socio-Technical Theory. *MIS Quarterly*, *1*(4), 11–28. https://doi.org/10.2307/249019

Braunstein, M. L. (2018). *Health Informatics on FHIR: How HL7's New API is Transforming Healthcare*.

Bygstad, B. (2015). The Coming of the Lightweight IT. *ECIS 2015 Completed Research Papers*, 1–16.

Bygstad, B., Stople, A., Steinsund, H., & Iden, J. (2017). Lightweight It and the It Function: Experiences From Robotic Process Automation in a Norwegian Bank. *Bibsys Open Journal Systems*, *25*(1), 11.

Christensen, B. H. (2018). *Anskaffelse og Implementering av forretningssystemer*.

Cyberedge Group. (2019). *2019 Cyberthreat Defense Report*. Retrieved from https://www.imperva.com/resources/reports/CyberEdge-2019-CDR-Report-v1.1.pdf

DiCicco-Bloom, B., & Crabtree, B. F. (2006). The qualitative research interview. *Medical Education*, *40*(4), 314–321. https://doi.org/10.1111/j.1365-2929.2006.02418.x

Everhard, J. (2019). The Pros And Cons Of Citizen Development. *Forbes*, p. 1. Retrieved from https://www.forbes.com/sites/johneverhard/2019/01/22/the-pros-and-cons-of-citizen-development/#41b8c64c84fd

Feldman, D. (2013). Drag & Drop: Think Twice. Retrieved December 18, 2019, from Medium website: https://medium.com/@dfeldman/drag-drop-think-twice-49e7bf3e6b31

Gartner. (n.d.). Citizen Developer. Retrieved from https://www.gartner.com/en/information-technology/glossary/citizen-developer

Haugnes, G. M. (2018). Å skille lettvekt-IT og tungvekt-IT setter fart på digitaliseringen. Retrieved December 18, 2019, from Titan UIB website: https://titan.uio.no/node/2723

Heggernes, T. A. (2017). *Digital Forretningsforståelse: Fra Store Data til Små Biter* (2nd ed.). Bergen: Fagbokforlaget.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly: Management Information Systems*, *28*(1), 75–105.

Hossain, S. (2018). Challenges of Software Quality Assurance and Testing. *International Journal of Software Engineering and Computer Systems*, *4*(1), 133–144.

ISTQB. (n.d.). ISTQB Glossary. Retrieved December 19, 2019, from https://glossary.istqb.org/en/search/

Johnson, F., & Eide, S. S. (2018). *An Inquiry Into Robotic Process Automation Implementation in Institutions for Higher Education*.

Konschake, M. (2018). Work in Progress and the Importance of Speed in Software Delivery. Retrieved December 18, 2019, from Medium website: https://medium.com/@Infinite_Monkey/work-in-progress-and-the-speed-in-software-delivery-e549a6fdb7f0

Korolov, M. (2019). 4 security concerns for low-code and no-code development. Retrieved from CSO website: https://www.csoonline.com/article/3404216/4-security-concerns-for-low-code-and-no-code-development.html

Kovalev, A. (2018). The Robotic Process Automation. How no-code/low-code prefix can attract new value-added. Retrieved December 18, 2019, from Medium website: https://medium.com/@andreykovalev_82500/the-robotic-process-automation-how-no-code-low-code-prefix-can-attract-new-value-added-66cc8ac3788

Kratzke, N. (2014). Lightweight Virtualization Cluster How to Overcome Cloud Vendor Lock-In. *Journal of Computer and Communications*, *02*(12), 1–7. https://doi.org/10.4236/jcc.2014.212001

Kunnskapsdepartementet. (2017a). *Digitaliseringsstrategi for universitets- og høyskolesektoren*. Retrieved from https://www.regjeringen.no/no/dokumenter/digitaliseringsstrategi-for-universitets--og-hoyskolesektoren---/id2571085/

Kunnskapsdepartementet. (2017b). *IKT- strategi for administrative tjenester: 2017-2021*.

Lucas, H. C., & Goh, J. M. (2009). Disruptive technology: How Kodak missed the digital photography revolution. *Journal of Strategic Information Systems*, *18*(1), 46–55. https://doi.org/10.1016/j.jsis.2009.01.002

Lunsford, D. L., & Collins, M. R. (2008). The CRUD Security Matrix: A Technique for Documenting Access Rights. *Annual Information Institute Conference Proceedings*. Retrieved from www.security-conference.org

Martin, James. (1982). *Application development without programmers*. Englewood Cliffs, N.J: Prentice-Hall.

Martin, Jeffrey. (2018). Impact of Low Code App Development on Testing. Retrieved December 20, 2019, from Smartbear website: https://smartbear.com/blog/test-and-monitor/impact-of-low-code-app-development-on-testing/

Marvin, R. (2014). How low-code development seeks to accelerate software delivery. *SD Times*, 1. Retrieved from https://sdtimes.com/application-development/low-code-development-seeks-accelerate-software-delivery/

Matthews, K. (2019). How Digital Transformation Changes Security Needs. Retrieved December 18, 2019, from Information Age website: https://www.information-age.com/digital-transformation-changes-security-needs-123478114/

Misra, S. (2018). Low-Code Application Development: A Catalyst to Digital Transformation. Retrieved from DevOps.com website: https://devops.com/low-code-application-development-a-catalyst-to-digital-transformation/

Morgan, S. (2016). Global Cybercrime Damages Predicted To Reach $6 trillion annually by 2021. Retrieved December 17, 2019, from Cybercrime Magazine website: https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/

Murphy, K. (2018). Robotic Process Automation and Low-Code. Retrieved December 18, 2019, from OutSystems website: https://www.outsystems.com/blog/posts/robotic-process-automation-low-code/

Newman, L. H. (2018). How Hackers Slipped By British Airways' Defenses. Retrieved November 9, 2018, from Wired website: https://www.wired.com/story/british-airways-hack-details/

NHH. (n.d.-a). SECTION FOR ADMISSIONS. Retrieved December 18, 2019, from Norges Handelshøyskole website: https://www.nhh.no/en/about-nhh/organisation/administrative-offices/office-of-student-and-academic-affairs/section-for-admissions/

NHH. (n.d.-b). STRATEGI FOR NHH 2018 - 2021. Retrieved December 18, 2019, from Norges Handelshøyskole website: https://www.nhh.no/contentassets/d8defdddb1c944acbbeebf1f1354cd52/nhhs-strategi-2018-2021.pdf

NIST. (2002). *The Economic Impacts of Inadequate Infrastructure for Software Testing*. Retrieved from https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf

NTNU. (n.d.). BOTT-samarbeidet. Retrieved from https://innsida.ntnu.no/wiki/-/wiki/Norsk/BOTT-samarbeidet

Osmundsen, K., Iden, J., & Bygstad, B. (2017). *HVA ER DIGITALISERING , DIGITAL INNOVASJON OG DIGITAL TRANSFORMASJON ? EN LITTERATURSTUDIE*. Bergen.

OutSystems. (n.d.-a). Entities. Retrieved November 12, 2019, from https://success.outsystems.com/Documentation/11/Developing_an_Application/Use_Da ta/Data_Modeling/Entities

OutSystems. (n.d.-b). How does OutSystems support testing and quality assurance? Retrieved December 20, 2019, from https://www.outsystems.com/evaluation-guide/how-does-outsystems-support-testing-and-quality-assurance/

OutSystems. (n.d.-c). OutSystems Security Overview. Retrieved from https://www.outsystems.com/evaluation-guide/outsystems-security-overview/

OutSystems. (n.d.-d). Static Entities. Retrieved from https://success.outsystems.com/Documentation/11/Developing_an_Application/Use_Data/Data_Modeling/Static_Entities

OutSystems. (2019a). *The State of Application Development: Is IT Ready for Disruption? 2019/2020*.

OutSystems. (2019b). *The State of Application Development*. 1–20.

OutSystems. (2019c). What do Gartner and Forrester say about OutSystems? Retrieved from https://www.outsystems.com/evaluation-guide/what-do-gartner-and-forrester-say-about-outsystems/

Pemberton, C. (2016). What Makes a Marketing Center of Excellence? Retrieved December 18, 2019, from Gartner website: https://www.gartner.com/en/marketing/insights/articles/what-makes-a-marketing-center-of-excellence

Porter, M. E., & Heppelmann, J. E. (2014). How Smart, Connected Products Are Transforming Competition. Retrieved December 18, 2019, from Harvard Business Review website: https://hbr.org/2014/11/how-smart-connected-products-are-transforming-competition

RapidAPI. (2019). API Key – What is an API Key? Retrieved from https://rapidapi.com/blog/api-glossary/api-key/

Revell, M. (2019). What Is Low-Code? [2019 Update]. Retrieved December 18, 2019, from https://www.outsystems.com/blog/what-is-low-code.html

Richardson, C., & Rymer, J. R. (2014). *New Development Platforms Emerge For Customer-*

Facing Applications. Retrieved from Forrester Research database

Richardson, C., & Rymer, J. R. (2016a). The Forrester Wave ™ : Low-Code Development Q2. In *Forrester*.

Richardson, C., & Rymer, J. R. (2016b). Vendor Landscape: The Fractured, Fertile Terrain Of Low-Code Application Platforms. *Forrester*. Retrieved from Forrester Research database

Ross, M. (2018). 4 essential features of modern low-code development platforms. Retrieved December 18, 2019, from InforWorld website: https://www.infoworld.com/article/3287146/4-essential-features-of-modern-low-code-development-platforms.html

Rotter, S. (2019). The Low-Code Development Market in 2019. Retrieved December 17, 2019, from OutSystems Blog website: https://www.outsystems.com/blog/low-code-development-market.html

Rymer, J. (2018). *Appian World 2018: Forrester - The Future of Low-Code Development*. Retrieved from https://www.youtube.com/watch?v=URimrHPBfeU

Rymer, J. R. (2018). Why You Need To Know About Low-Code, Even If You're Not Responsible For Software Delivery. Retrieved from Forrester website: https://go.forrester.com/blogs/why-you-need-to-know-about-low-code-even-if-youre-not-responsible-for-software-delivery/

Samordnet Opptak. (2008). Om Samordna opptak. Retrieved December 19, 2019, from UNIT website: https://www.samordnaopptak.no/info/om/

Siemens. (2019). *Digital Temperaturmåler for Norsk Næringsliv 2019*. Retrieved from https://assets.new.siemens.com/siemens/assets/public.1565082616.7d1de681-d583-47b6-af45-08cf3d8befbe.digital-temperaturmaaler2019.pdf

Simon, H. A. (1970). The Sciences of the Artificial. In *Technology and Culture* (Vol. 11). https://doi.org/10.2307/3102825

Sommerville, I., Cliff, D., Calinescu, R., Keen, J., Kelly, T., Kwiatkowska, M., … Paige, R. (2012). Large-Scale Complex IT Systems: The reductionism behind today's software-engineering methods breaks down in the face of systems complexity. *Communications-*

Acm, *55*(7), 71–77. Retrieved from http://uwashington.worldcat.org.offcampus.lib.washington.edu/title/large-scale-complex-it-systems-the-reductionism-behind-todays-software-engineering-methods-breaks-down-in-the-face-of-systems-complexity/oclc/800275499&referer=brief_results%5Cnhttp://dl.a

Stroud, F. (n.d.). Shadow It. *Webopedia*. Retrieved from https://www.webopedia.com/TERM/S/shadow-it.html

Tanna, D. (2016). A brief Note about Application Programming Interface. Retrieved December 18, 2019, from emiprotechnologies website: https://www.emiprotechnologies.com/blog/setu-65/post/a-brief-note-about-application-programming-interface-375

Taulli, T. (2019). What You Need To Know About The Low-Code Market. Retrieved December 18, 2019, from Forbes website: https://www.forbes.com/sites/tomtaulli/2019/02/17/what-you-need-to-know-about-the-low-code-market/#75bd5a935afc

Terhorst-North, D. (2013). ARE WE NEARLY THERE YET? Retrieved December 18, 2019, from DAN NORTH & ASSOCIATES website: https://dannorth.net/2013/07/05/are-we-nearly-there-yet/

Tiemens, S. (2019). The low-code journey: from experimenting and discovering to delivering value at scale and continuous improvements. Retrieved from https://home.kpmg/nl/nl/home/insights/2019/07/the-low-code-journey-from-experimenting-and-discovering-to-delivering-value-at-scale-and-continuous-improvements.html

Tschabitscher, H. (2019). The Outlook.com SMTP Settings You Need to Send Email. Retrieved December 19, 2019, from Lifewire website: lifewire.com/what-are-the-outlook-com-smtp-server-settings-1170671

Ulfsten, A., & Iden, J. (2018). For mye uavhengighet. *Stat Og Styring*, *4*, 44–47.

Unit. (n.d.). Felles studentsystem. Retrieved from https://www.fellesstudentsystem.no/

Urquhart, C., & Ravindranathan, M. (2006). Management Information Systems (G. M. Marakas, Ed.). *Journal of Documentation*, 8th ed., Vol. 62, pp. 534–535. https://doi.org/10.1108/00220410610673882

Vaishnavi, V., Kuechler, B., & Petter, S. (2019). *Design Science Research In Information Systems*. (1), 1–66. https://doi.org/1756-0500-5-79 [pii]\r10.1186/1756-0500-5-79

Van Vleck, T. (2001). The History of Electronic Mail. Retrieved from https://www.multicians.org/thvv/mail-history.html

Vial, G. (2019). Journal of Strategic Information Systems Understanding digital transformation : A review and a research agenda. *Journal of Strategic Information Systems*. https://doi.org/10.1016/j.jsis.2019.01.003

Vincent, P., Lijima, K., Driver, M., Wong, J., & Natis, Y. (2019). Magic Quadrant for Enterprise Low-Code Application Platforms. Retrieved December 18, 2019, from Gartner website: https://www.gartner.com/doc/reprints?id=1-1FKNU1TK&ct=190711&st=sb

Warren, N. (2018). Low-Code Myths, Fears, and Realities: Security. *OutSystems*, 1. Retrieved from https://www.outsystems.com/blog/posts/low-code-myths-security/

Wayner, P. (2019). Why developers hate low-code. Retrieved December 20, 2019, from InfoWorld website: https://www.infoworld.com/article/3438819/why-developers-hate-low-code.html

White, S. A. (2004). *Introduction to BPMN*. *15*(1_suppl), 1–2. https://doi.org/10.3727/000000006783982421

Wilfrid, D. (2018). A Brief History of Low-code Development Platforms. Retrieved December 18, 2019, from Quickbase website: https://www.quickbase.com/blog/a-brief-history-of-low-code-development-platforms?fbclid=IwAR1uBYG4NxN_YmHY02fDfewNujG2gti8UXRsl0p3kqcjgHAJoQWd67ikZ3s

Yoo, Y., Henfridsson, O., & Lyytinen, K. (2010). Research Commentary—The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research.

Information Systems Research, *21*(4), 724–735. https://doi.org/10.1287/isre.1100.0322