Beacon
Project Main Report

Version 4.0

Revision History

Date	Version	Description	Author
27. March 2023	1.0	First draft	L. C. W. Aune, B. E. Christiansen and J. D. Eriksen.
16. April 2023	2.0	Second draft	H. R. Billingstad, V. R. Ellingsson, L. C. W. Aune, B. E. Christiansen and J. I. D. Eriksen.
26. April 2023	3.0	Third draft	H. R. Billingstad, V. R. Ellingsson, L. C. W. Aune and B. E. Christiansen.
27. April 2023	4.0	Final project report	H. R. Billingstad, V. R. Ellingsson, L. C. W. Aune, B. E. Christiansen and J. I. D. Eriksen.

Abstract

During the spring semester of 2023, the primary focus in the course IDATT1002 Systemutvikling 1 has been team collaboration on a larger project assignment. Honouring the requirements of the project, the team has developed an application that handles both budgeting and accounting. The team paired up with an external client, who through a series of meetings and usability tests has played a significant role in the development of the application. The final product is named Baecon after the client's homestead Bærum in combination with the application's purpose as an economic tool.

The team chose Scrum as the system development method, as it is agile and provides structure and progression in the project. Taking this into consideration, the semester has consisted of Scrum sprints, guidance meetings, development of new software, usability tests and documentation. The project has given the team valuable experiences in system development, handling a large number of feedback, in addition to developing new software that considers principles of universal design.

At the end of the semester, the team had developed a functioning application with a number of features, including the ability to attach documents to accounting entries and accommodate for colour blindness and screen reading. While the team acknowledges the possibility for improvements in future iterations, like extending the time range that the application supports, the team remains with valuable experiences from the project.

Preface

In this report, we present our learning journey throughout the course IDATT1002 for software development students at Norwegian University of Science and Technolgy. The course has been designed to promote learning in several fields within software development. Hereunder, working as a team through agile software development, user testing and interaction design. To achieve this the students were assigned a project to which they could apply the taught theory in a simulated real-world scenario. By doing so, students were compelled to work as a team and leverage each other's strengths while also learning from one another.

We would like to express our gratitude to our course instructor, Associate Professor Mohammed Ali Norozi, for his support, guidance, and invaluable insights throughout the project. His thoughts and suggestions have proved most important to our work. Furthermore, we would like to give a special thanks to our Student Assistant Daniel Andre Evensen who has contributed with insightful guidance, with a keen eye for system development. Lastly, we would like to direct a heartfelt thanks to Mr. Svein Magnus Walaas for fulfilling the role as our client in this project. His feedback and collaboration have been instrumental in shaping our software and driving it towards the product that it is today.

This report, and the software development project it refers to, represent the culmination of our team members' hard work and dedication. Throughout the project, we have collaborated extensively to bring it to its full potential. In doing so, we have not only gained a deeper understanding of software development principles but also fostered personal growth by embracing teamwork and overcoming challenges. As we reflect on our journey, we are grateful for the invaluable lessons learned and the unique experiences that have shaped us as future professionals in the field of software development.

Lotte Aune — Vemund E. Røe

Håkon R. Billingstad — Birthe Emilie Christiansen

John I. Eriksen — Trondheim
27/04-23

Table of Contents

List of Figures

Assignment description

This assignment was given to computer engineering students at NTNU in Trondheim. The students were given the task of finding a client for whom they were to develop a finance software system. Very few guidelines and constraints were given, other than that the client could not be a student within the same or similar program of studies, nor be working within the field of computer technology.

The group decided to ask a Renewable engineering student at NTNU as a client. This person was chosen because of the friendly relationship he has with two of the group’s members. Furthermore, the client has a genuine interest in both finance and computer technology but no formal education in either of the fields.

To learn about system development, working as a team, and to simulate real circumstances, the students were given few suggestions on how to execute the assignment. Therefore, the team has tried to integrate as much theory on system development into the process as possible. This also enhances the learning experience. To achieve success, the team manufactured a vision document that outlined clear expectations, risks and goals for the software. These documents are attached in the appendix of the report.

Chapter 1 Introduction

This report marks the end of a project developed for the IDATT1002 course at the NTNU. The project aimed to create a tool that provides private individuals or small businesses with an overview of incomes and expenses, and thus empower them to improve their economy. The project is now completed, achieving most of the objectives set out at the beginning. This report serves as final documentation of the project, providing a detailed overview of the development process and outcomes that followed.

The report begins with an abstract which summarizes the report, followed by a preface that describes the purpose of the project. Afterward, there is an overview of the report's content, and a description of the assignment. Chapter 1 is a brief introduction to the report, which includes a dictionary of relevant definitions, acronyms, and abbreviations. The main part of the report starts in Chapter 2, where the theoretical background for the report is presented. This chapter provides the essentials of the GDPR and Professional ethics, UML and modelling, User Testing and Interaction Design. Chapter 3 describes the decision-making process and the choices made, based on the theory presented in Chapter 2. This chapter also includes a description of how the team organized themselves during the project period and offers an overview of the literature, the software used in the project, as well as the documentation. Chapter 4 presents both the project results and the administrative results. In Chapter 5, a discussion and analytic evaluation of why the project turned out as it did is provided. The process of the project, handling the various feedbacks and accomodation for universal design makes themselves relevant in this chapter as they are discussed in respective subchapters. The main part of the report concludes with Chapter 6, where the key findings are summarized, and recommendations for future improvements are made. The report ends with a repository, a list of references and an appendix.

Definitions, Acronyms and Abbreviations

JDK	–	Java Development Kit
LTS	–	Long Term Support
MVP	–	Minimum Viable Product
UI	–	User Interface
GUI	–	Graphical User Interface
GDPR	–	General Data Protection Regulation
UML	–	Unified Modelling Language
UX	–	User Experience
WCAG	–	Web Content Accessibility Guidelines
WAD	–	Web Accessibility Directive
Wireframe	–	Low-resolution visual prototype of a graphical interface
IA	-	Information Architecture

Chapter 2 Theory and relevant literature

GDPR and professional ethics

The General Data Protection Regulation, or GDPR for short, is a regulation on how personal data should be stored and processed. The regulation applies within the European Union, EU, and the European Economic Area, EEA, and it establishes a set of rules that organizations must follow when processing data of EU and EEA citizens. The aim of the regulation is to strengthen individuals' rights regarding their personal data and to increase transparency in how organizations handle data.[1] These individual rights include the right to access personal data, the right to have your data erased, the right to have your data corrected, and the right to know how your data is being used.[2]

Both NITO and Tekna provide guidelines to ensure that engineers work according to ethical standards. NITO's guidelines aim to promote an inclusive and respectful work environment, while also emphasizing the importance of ethical considerations and quality standards. This involves being respectful of co-workers and business partners, upholding professional responsibility by following quality standards, and being aware of how technology and human values interact.[3] Tekna's guidelines are more focused on personal accountability for own choices. Members are required to make decisions that are moral, fair and honest and do not negatively impact people, the environment, or society. For example, an engineer developing an application should avoid using deceptive design practices that trick users into making purchases or agreeing to different conditions.[4] By following NITO's and Tekna's guidelines, engineers are well on their way to ensure that an application has been developed according to ethical standards.

UML and modelling

Unified Modeling Language, UML, is a standardized visual language for modelling software systems, business processes, and organizational structures. It provides a consistent way to visualize and communicate the design of a system using various types of diagrams, such as use case, class, object, sequence, activity, state, component, and deployment diagrams. UML is managed by the Object Management Group and is widely used by professionals within software engineering, systems analysis, and related fields for designing, developing, and maintaining complex systems.[5]

UML consists of several diagrams, some examples are use case, class, sequence, and activity diagrams. Use-case diagrams show high-level interactions between a system and its users, capturing functional requirements. Class diagrams represent the static structure of a system, displaying classes, attributes, operations, and relationships. Sequence diagrams illustrate the flow of messages and the order of events in a system. Activity diagrams describe the dynamic aspects of a system by modelling the flow of control between activities, often used for business processes and workflows. These diagrams are essential tools for visualizing and communicating various aspects of a system's design.

[1] Personopplysningsloven 2022

[2] Datatilsynet 2018.

[3] NITO 2021.

[4] Sandstrak 2023: 8.

[5] Tesdal 2020a: 4.

Wireframes is another form of commonly used modelling before starting with initial software development. It is used to demonstrate a visual representation of a page and often include minor functional availabilities to test for intended behaviour of the program.[6] This works as a tool for initial user presentation as it does not require extensive development. Therefore, wireframe models are both time and cost effective which also gives software developers a head start in user testing. Since most wireframes are low-resolution models, they are ideal for trying out different versions of a user interface and can be revised and tested in quick succession.

User testing

User testing is a crucial part when developing software, as it allows developers to gather valuable feedback and insights from users and the customer at different stages of the project. This feedback and insight help in making decisions regarding the further development of the software. Performing user testing in different stages of the project, such as the wireframing phase, after developing the MVP and before the release of the software, ensures that the software is refined and optimized to meet the user expectations.[7]

Testing wireframes is the initial step in the user testing process, where developers assess the application’s basic structure, layout and navigation. At the wireframe stage, user testing focuses on the overall usability and flow of the application. It helps in identifying any potential navigation or layout issues that may hinder user experience. By testing wireframes, developers can make adjustments before investing resources in developing the software. As the project progresses and the development of the MVP is done, the user testing shifts focus on the functionality, performance and integration of features. Testing the MVP can help address functionality issues and collect user feedback on the overall experience. Finally, user testing before the release of the product ensures that the application is fully functional and ready for the customers.

Interaction design

As computers and modern information technology in one form or another has become an unavoidable part of everyday life for most people, the need for well thought-out and easy-to-use design have become increasingly important. Already in 1983, Professor Don Norman who is renowned for his books within principles of design, outlined a set of design principles that are still considered fundamental knowledge for design engineers.[8]

This field of study is known as Interaction Design, ID. It deals with all the peculiarities of how humans perceive and interact with a vast number of different interfaces, and strives to work out good solutions for both easy and powerful interactions. An interface is, in its simplest terms, merely the boundary where two things meet, through which one system, for example a human, interacts with a system, for example a computer. There are many design layers, but a few that stands out are the User Experience, Information Architecture, Usability and User Interface design. They are concerned with the structuring and layout of both

[6] U.S. General Services Administration 2023.

[7] Kathayat 2023: 41.

[8] Norman 1983.

interface elements and information placement, and aims to make interacting with the product an experience that is as positive as possible. The placement and design of all visible elements, as well as how the information is organised have an impact on how efficiently users are able navigate and use the interface, and thus operate the machine being interacted with. Additionally, designers strive to enhance the experience itself. The experience should be one where the user is left with a positive feeling about the interaction, be it with an app or a car's centre console.

Accessibility and Universal Design have become a natural focus for both governments and companies, which is evident in the fact that there are now emerging several governmental regulations for both. This is illustrated by the Norwegian Parliament's decision to require applications and websites to be compliant with the European Union's Web Accessibility Directive guidelines by 1 February 2023.[9] Accessibility focuses on the principle that everyone should be able to use modern technology. These requirements are accompanied by well-established guidelines for how to implement these designs, such as Web Content Accessibility Guidelines and the more recent WAD.[10]

To comply with WCAG 2.1 Principle 1 - Perceivable, applications must meet a number of key requirements.[11] To provide alternative text for non-text content, 1.1.1, offer alternatives for time-based media, including captions and audio descriptions, 1.2.1, 1.2.2 and 1.2.3, ensure programmatically determined information, structure, and reading sequence, 1.3.1 and 1.3.2, avoid relying solely on sensory characteristics for instructions, 1.3.4, and not use colour as the only means of communication, 1.4.1. Additionally, the application must have a mechanism to pause or stop automatically playing, 1.4.2.

Familiarity also plays a huge role in how users interact with a given application. As usability design expert doctor Jakob Nielsen outlined in his *10 Usability Heuristics*, design that relies on the user feeling familiar, recognizing workflows and being empowered via something as simple as an undo and redo function, will likely be successful.[12] While there may be fancier ways of presenting information, there's something to be said for the familiar and functional. Even when users express a desire, there's abundant evidence that supports the fact that the generic user usually «don't really know what they want».[13] This has nothing to do with intellectuality, but simply that most users do not really have neither the experience nor insight required to make good and functional design solutions. This is why user testing is so crucial. Observing the users and how they really interact with a product, rather than having them tell you a story how they might see themselves interact with it.

[9] Digitaliseringsdirektoratet 2023a; Forskrift om universell utforming av IKT-løsninger 2023: §4.

[10] Digitaliseringsdirektoratet 2023b.

[11] Kirkpatrick, Connor, Campbell & Cooper 2018.

[12] Nielsen 2020.

[13] Nielsen 2001; Moelk 2018.

System development methodology

System development methodology is a relevant term for teams that jointly develop and deliver software. It draws several parallels to the «Godfot»-theory of the late soccer coach Nils Arne Eggen, which highlights the importance of teamwork in order to succeed. This is highlighted in his first postulate, where it is stated that being proficient is not a solo performance.[14] In particular the system development methodology conveys something about how a team could work efficiently and at the same time maintain quality in the final delivery.

The Waterfall Method, often denoted as one of the traditional methods, was the dominant system development method from the 1970s and onwards.[15] The method consisted of a number of finite and well documented processes, delegated according to roles and function. The fact that one process was followed by the next resembled the concept of relays. The rigid documentation requirement was used to ensure that necessary knowledge was transferred to the next team.

From the mid-1990s it is possible to trace a shift away from the prevailing traditional methods. These were termed as the agile methods. The reaction came in particular as a result of the overwhelming requirement to document.[16] The 2004 *Agile Manifesto for Software Development* prioritizes adapting to change, with agile methods defined by their iterative and incremental nature.[17] They adopted the significant steps from the Waterfall Method but repeats them over and over again in several iterations. Unlike the traditional methods, the Agile methods emphasize the learning of problem-solving that takes place in the development of a project. As a result of its iterative character, the agile methods will be more flexible to change in the face of the need, as this can be detected and handled at an earlier stage compared to the traditional methods.

There are a number of methods that can be characterized as agile. Kanban, Extreme Programming and Scrum can in this context be cited as illustrative examples. The latter one is widely used because it is suited for developing new software and maintaining existing software.[18] Scrum projects are suitable for smaller teams consisting of up to seven members, as it is based on a flat structure. This applies with the exception of the product owner, who is responsible for the communication with the client and the scrum master, who facilitates the daily orientational stand-up meetings. Scrum distinguishes itself by intensifying iterations in short sprints, usually lasting a couple of weeks that should lead to deliverables using clear goals. Each Scrum sprint is concluded with a sprint-review, which emphasizes the position of learning in the agile methods.[19] The review opens for assessing the experiences from the former sprint and discusses what could be changed for the following one.

[14] Eggen 1999: 207.

[15] Tesdal 2020b: 3.

[16] Cohen, Lindvall & Costa 2004: 3.

[17] Beck, et al. 2001.

[18] Tesdal 2020b: 12; Cohen, Lindvall & Costa 2004: 14.

[19] Tesdal 2020b: 8.

Chapter 3 Method

Methodology

In this project, Scrum has been chosen as the appropriate system development method. This choice can be justified on the basis of several arguments. Firstly, it can be justified due to the shift from the traditional and heavily documented methods to more agile ones. Furthermore, this can be rationalized by the fact that Scrum has a wide spread among software developers, and will therefore contribute valuable experiences for the team members. Based on the project's purpose of developing an application for a client, Scrum may be considered more convenient for this project as it is stated as better suited for projects developing new software. If, on the other hand, the purpose of the project would have been to maintain the software of an existing application, Kanban would run as a suitable method candidate.

A second argument for choosing Scrum is based on the project's progress divided into three distinct iterations, vision, MVP and final delivery. Each of them is defined with a finite end in the form of a guidance meeting with lecturer or a delivery. The team interpreted these iterations as Scrum sprints, lasting three to four weeks each. This choice made it natural to set clear interim goals leading up to the deliveries. In the longer term, this contributed to progression in the work with the application. Besides the two guidance meetings with the lecturer, the team conducted weekly meetings with the assigned student assistant and internal team meetings in the form of stand-ups. During the stand-ups, which took place at the beginning of team meetings, communication around progression, challenges and work distribution was facilitated. Each team member had the opportunity to give their inputs and seek assistance when needed. In accordance with the collaboration agreement, a shorter or longer memo was written after each meeting by the team's selected referent, depending on the status of the meeting as formal or informal. Beyond documenting and structuring the work put into the project, these notes also formed the basis for the sprint reviews that marked the end of each sprint. In this way, experiences and feedback from the previous sprint were taken into account before the next one started.

Work and role distribution

The scrum method's role distribution may appear as a final argument for the justification of scrum as the chosen method for this project. The team leader was chosen to act as the product owner with the responsibility to solely communicate with the client, due to professionalism and orderliness when facing the client. One of the other team members was chosen as the Scrum Master and was therefore assigned with the overall responsibility for the source code, delegating assignments and facilitating the stand-ups. Beyond defining these roles, and in accordance with the collaboration agreement, the team chose to go for a flat hierarchy.

It was sought that everyone could participate in all the steps that culminated in the final product, and in this way increase competence in a broader field. This included everything from the team meetings in the start of the project with modulation and documentation of the visions, programming of the application and documentation of the development, which marked the end of the project. The team sought to work for shorter work periods with clear tasks defined as separate issues on the issue board, in an attempt to keep up with the scheduled sprints. The distribution of tasks was either agreed upon by the stand-ups or already determined in the employment contract. For instance due to the sprint deadlines, there was a need to divide the team into smaller units. This inner division was particularly evident in connection with the first sprint, where the establishment of the project’s visions was determined through an internal discussion within the team. Three team members were assigned to formulate the vision document, while the remaining two members were tasked with modulating required UML diagrams and a wireframe. In addition to the goal of reaching the deadline, this choice was also made on the basis that the entire team could start developing the application at the same time. In this way, the Scrum Master could delegate issues and assist with support, whilst programming parallel. As the MVP started to resemble a functioning product, and several high-priority issues checked off the issue board, team members were eventually assigned to document the process of the project.

Project requirements

In attempting to meet the project’s assigned requirements, the team determined on a number of decisions related to prototyping, usability testing and universal design. An important choice was prioritising the client from the beginning of the project. This culminated in a meeting with the client in the third week of the project. Through dialogue with the client, key visions for the application were discussed and drawn up. The team’s marked prioritization of the client could be justified by the fact that these inputs would have an impact on the outline of the final product. Although in the first instance as a basis in the early prototyping phase. The requirements specification, the client’s wishes and the team’s visions were altogether taken into account in the design of the UML diagrams and the Wireframes.

Correspondingly with the project’s three sprints, the team chose to conduct three rounds of usability tests, to ensure feedback throughout the development of the application. This included a test of the prototyped Wireframes, the MVP and the final product. In addition to the unit test of the source code, these tests were carried out to develop a robust and functioning product for the client. This can be rationalized with the fact that the feedback could be taken into consideration in the sprint review, and therefore could have an improving impact on the subsequent iteration. The team chose to also conduct a third and final usability test on the final product for two reasons. First, the test would provide a benchmark of the user experience measured against the initial visions for the application. Secondly, the test would also be able to attest to improvements that could have been implemented in the potential following sprint. Along with the client who participated in each test round, the team also strived to run the tests on a wider range of potential users. This choice can be justified by a desire to gather input from multiple perspectives.

In accordance with the project’s demand to develop an application that satisfies the first principle of WCAG 2.1, the team had to elaborate on how universal design would be implemented in the final product. The team decided on a number of measures to make the application accessible to as many people as possible. One of them was designing the application with regard to using familiar and expected conventions, such as ellipsis for implying something more happens when pushing a button. Furthermore, it was emphasized to make the application accessible for people who have vision impairments and need alternative navigation methods. The following measures have been chosen in an effort to facilitate universal design, capability for screen reading, colourblind mode and keyboard-based navigation.

Literature and Internet resources

For the theoretical part of the assignment, the lectures and course material provided by the lecturers served as the foundation for the theory chapter. To expand on this basis, websites like NNGroup, UUtilsynet and Lovdata stand out for providing insight in both theory, important principles, and regulations, especially with regards to usability and WCAG compliance.

For programming the application, the team built on knowledge gained through the curriculum assigned Java courses, IDATT1001 and IDATT2001. Baeldung and GeeksForGeeks provide coding tutorials by publishing examples for often asked questions, while StackOverflow focuses on users asking questions, which in turn is answered by other users to try to arrive at a solution. To aid in learning to use SceneBuilder, Youtube channels like “thenewboston” were used. Oracle.com provided documentation for methods and functionality in Java.

It is also worth mentioning that the team has used ChatGPT as a resource. Alongside the websites mentioned above, ChatGPT has been used for asking questions and getting solution proposals when stuck. If useful, the solutions provided have been adapted to fit the project. In its current state, ChatGPT is not able to generate code at the level of complexity used in the project application, and thus it is limited in its ability to replace a student's ability to code and do problem solving. The real value lies in getting suggestions that can then be adapted and built upon. The team members have used ChatGPT at their own discretion, some being more hesitant than others. That said, all team members used it to some degree, and the group as a whole trusts in the personal integrity of each individual member to not have used it uncritically or in a manner that would cast a shadow over the project and the team.

Software and tools

Several different third party tools and solutions were used for managing the project. Some were specified in the project assignment, while others were up to each team to decide for themselves. One requirement was for the team to create a repository on NTNU's GitLab, and use git for managing version control and storing source code, as well as using Issue Boards and GitLab Pages. The Issue Boards were used to manage, assign and track work tasks. Issues were created and marked with tags according to their priority and category. Gitlab Pages provide a way to publish JavaDocs for the application as a website, making them available outside inspecting the source files. Balsamiq Wireframes was used for creating the initial wireframe model, and DrawIO and JetBRains IntelliJ was used for modelling UML diagrams.

Other significant tools were the Microsoft Office suite, especially Outlook, OneDrive, Word and Excel. Outlook was used for issuing meeting notices for the formal meetings, while OneDrive, Excel and Word was used for storing, sharing and collaboration of project documents. The team chose to keep communications mainly through Facebook Messenger.

The application was written in Java, using JDK 17 LTS, and the IDE of choice for the team was JetBrains IntelliJ IDEA. The project assignment required the use of JUnit for Unit Testing, JavaFX for GUI and Maven as build tool. The GUI was made in JavaFX, using SceneBuilder.

Documentation

The documentation for this project have been made available in the form of a dedicated Wiki on GitLab. The Wiki contains information regarding the project and code structure, like strcture overview, unit testing, persistence, resources and relevant system UML diagrams.

To document the design iteration, a page for Wireframes and design have been made, with screenshots comparing the milestone designs from Wireframe to MVP to Final Delivery. This is complemented by a page focusing on Universal Design, to explain how the thinking and process for make the application adhere to Accessibility and Usability standards.

To aid in the setup and use of the application, here are dedicated pages for user manual and installation manual.

Chapter 4 Results

Project results

There were seven product features planned to be implemented, as described in Chapter 6 and Chapter 9 of the vision document. Additionally, two more features were added subsequently, after the team agreed upon their importance during project development. These are the universal design feature and the dashboard page feature. Below is a table that displays the final status of each product feature, along with a description and ranking of its importance.

Table 1 Product features

Feature	Description	Importance	Achieved
Budget overview	A budget overview which shows both monthly and annual budgeted income and expenses. It should be possible to switch between a monthly or annual view. The data is represented with both cold, hard numbers, and by easy-to-digest visual methods such as pie charts, bar charts and trend line charts.	Highest	Yes
Accounting overview	A view, separate from the main budget, where expenses can be added. There is an option to append files, such as a PDF, to an entry. The expenses get added on a rolling basis, and the total spending appears in a summary.	Highest	Yes
Recurring payments	Ability to add a payment and define it as recurring. It’s possible to choose between the payment repeating weekly, monthly, or yearly.	Medium / Nice to have	Yes
Visualized data	The user’s financial situation should be shown “at a glance” via different charts.	Medium / Nice to have	Yes
File archive	Overview of all files appended to an entry, with relevant information to identify the entry and content. The overview has a section for miscellaneous documents	Important	Partly

	not tied to a specific entry.		
Dark mode	Dark colours in the interface, to reduce eye strain.	Low / Nice to have	No
Standalone desktop application	The application is an offline application that runs locally on the user’s laptop or desktop computer. The database is password protected, and it’s possible to export the database, or load/import data from an external database.	Medium / Nice to have	Partly
Universal Design	The application must be designed according to WCAG 2.1 principle 1 - Perceivable	Highest	Yes
Dashboard page	A dashboard which shows data from both the budget and the accounting. The data is represented with both numbers and charts.	Highest	Yes

In total 12 user tests were conducted throughout the product development process. Specifically, there were three user tests of the wireframes, seven user tests of the MVP and two user tests of the final product, with the client participating in one test for each of the stages. The feedback from the user tests were used to evaluate the significance of a feature, identify potential improvements, and guide future feature functionality decisions. For example, the budget overview and dashboard page underwent significant revisions after MVP User Test Participant 01 pointed out that “it is evident that the pages are made by different people”, and MVP User Test Participant 06 requested a “more consistent formatting”. Testers occasionally disagreed, as demonstrated by the Customer MVP User Test, in which the customer preferred to keep the layout of the Wireframes rather than achieving consistency by using the same kind of layout on every page.

The final test, conducted on Participant 01 and the client, provided valuable insights for selecting the colours of the charts in the application. Despite having partial red-green colour blindness, Participant 01 was able to differentiate between the bars, lines, and sectors on the charts with the suggested red and green theme. However, she found it more comfortable to read the charts with the colourblind mode enabled. The client expressed his preference for the green and red colours in the charts, which was an important consideration in the design process as the client’s preferences should be accommodated in the application.

The application's compliance with accessibility requirements was achieved through various methods. For instance, the 1.1.1 Non-text Content criterion was met by including a connected table view or text for each chart, enabling users with visual impairments to access the data. The 1.3.1 Information and Relationships requirement was addressed by ensuring compatibility with screen readers, and the 1.3.2 Meaningful Sequence requirement was met by presenting content in a logical and readable order. The 1.3.3 Sensory Characteristics criteria was addressed by using red font and a minus sign to represent negative numbers and maintaining a consistent design pattern across pages. Additionally, a colourblind mode was added to the application to meet the 1.4.1 Use of Colour requirement, enabling people with colour vision deficiencies to access information presented in charts. These measures show how the application was designed according to WCAG 2.1 principle 1 - Perceivable.

The aim of the project was to develop a tool that efficiently aids in managing personal finances and is consistent with the objectives stated in the vision document. Out of the nine features proposed, five were successfully implemented, while two were partly developed, and one was dropped. The file archive feature works as outlined in Table 1 but lacks the capacity to directly link receipts with corresponding entries. This limitation may pose a challenge when a substantial number of receipts are added, and the feature can only be considered partially achieved. Similarly, the standalone desktop application feature can also only be considered partially achieved, as the application does not store files in a password-protected database and does not allow for the importation or loading of data from an external database. Additionally, the dark mode feature was not included in the final product. Despite these limitations, the uncompleted features held a low to medium importance and did not have a significant impact on the primary functionality of the program. As a result, the final product aligns with the goals outlined in the vision document.

Administrative results

At the outset of the project, the team created a Gantt-diagram to establish deadlines and planned work time. The diagram illustrated the schedule, and for the majority of the tasks, the planned work time was scheduled to conclude approximately a week or a few days before the actual deadline. To account for unforeseen obstacles or setbacks, the team set an internal deadline and scheduled excess time spent as a buffer. The team mostly stuck to the Gantt-diagram and did not need to alter any of the task deadlines. Even in situations where the internal deadline had passed, it was generally not an issue as the internal deadline was a soft deadline and allowed for planned excess time spent.

As per the team’s collaboration agreement, three impact goals were set. These were “Get to know each other, build trust, and increase motivation”, “effective collaboration” and “Flexibility and solution-oriented behavior”. The first goal was to create a more relaxed and confident environment inside the team. During the product development process, the team frequently spent lunch breaks together and occasionally engaged in team-building activities like an escape room. These actions suggest that the team members were at ease with each other, and that the first goal was accomplished. The second goal focused on ensuring that each member could contribute according to their strengths while learning from one another. The timesheet summary reveals that each team member has participated in almost every category, although not everyone has worked the same number of hours in each category. This demonstrates that the team has worked in a manner that enabled members to learn from one another while also contributing to areas where they excel. The third goal aimed to encourage a flexible and solution-oriented approach to work. Through discussions during product development, team members have demonstrated a willingness to express their opinions and have encouraged all viewpoints to be presented. This indicates that the goal has been met. To summarize, the team successfully met all three impact goals.

The team worked on the project for a total of 757 hours, with each member contributing between 190,5 and 138,5 hours. In line with the “Standard Operation Procedures” outlined in the team’s collaboration agreement, work primarily took place during scheduled subject hours at school. This approach made it possible for most of the work to be done collaboratively on campus, although some subtasks were carried out independently. The activity that consumed the most hours was Coding, followed by Project Reporting. Additionally, the category requiring the most time investment was Documentation, succeeded by Beta Client (MMP). Despite dedicating more hours to the project than initially outlined, the team perceived this as a testament to the high success rate achieved in meeting the impact goals. Additionally, the team members spent several hours on campus not only working on the project but also engaging in social activities, fostering stronger team dynamics.

Chapter 5 Discussion

Process

The team began by drafting a collaboration agreement with terms that all members agreed to and signed. They committed to treating each other with respect according to known ethical standards of conduct in the workplace. Throughout the development process, team members were respectful and listened to each other's suggestions and concerns. When conflicts arose, they found suitable solutions that all parties could accept. Another key feature of the agreement was to work together by meeting at designated spaces on campus. A schedule for these meetings was also included in the agreement.

Although the team followed these guidelines for the most part, they initially failed to hold daily stand-up meetings as per the Scrum framework. This led to poor results in the first sprint, with tasks being duplicated and an unclear view of the project's progress. After experiencing communication issues and double work, the team recognized the importance of daily Scrum meetings and agreed to take more responsibility in ensuring their execution. Once these issues were addressed, the workflow improved significantly, and the team's efficiency increased as duplication and lack of project overview were eliminated. This also positively impacted the team's performance, understanding, and usage of the backlog and version control system in GitLab.

As daily Scrum meetings became more effective and team members gained experience with GitLab, the Scrum Master obtained better control of the development process. This allowed them to easily access the GitLab project and assign tasks based on Scrum meeting discussions. Consequently, team members experienced a higher degree of involvement, understanding their tasks and what others were working on. This led to a more demanding but manageable learning process. However, as the third and final sprint began, the team found the initial issue board ineffective. To address this, they spent a morning discussing how to overcome this challenge and conducted a thorough review of the GitLab backlog, categorizing uncompleted issues by priority. This allowed for better control and focus leading up to the final delivery of the software.

Overall, the team felt that the process was successful, with a steep but rewarding learning curve. While the lack of programming experience was a significant constraint, the team achieved more than expected. Time restrictions were also a challenge, causing some desired features to be deprioritized, such as "Dark mode" and a database for both password-protection and handling import and export of data. However, the team deemed developing a colourblind mode and mastering reader and writer classes more important for the learning process and usability of the program. As a result, the team sees the project as a tremendous success.

Feedback

The development of the final product has involved inputs and feedback that the team has had to consider. This included everything from the feedback given in the guidance meetings with the student assistant and the lecturer, to the diverse inputs from the usability tests. In sum, this culminated in a great number of considerations, and accordingly a couple of conflicts arose as the team tried to navigate through them. Among other things, these included discrepancies between initial vision for the development and input from the student assistant, as well as conflicting feedback from the usability tests.

First and foremost, the handling of the feedback relates to the choice of system development method for this project. It would be beneficial to argue the positive impact that the Scrum has had to help the team navigate through the feedback, while at the same time fulfilling the initial vision for the application. This can be justified by the fact that the stand-up meetings and sprint reviews have facilitated the discussion of these inputs continuously in each sprint. The method's characterisation as willing to change can also be highlighted. Scrum enabled the team to address and respond to the conflicting feedback continuously within each sprint, and therefore conduct measures when necessary. The traditional Waterfall method would on the other side act as an opposite based on the methods streamlined practices. While Scrum allowed rapid responses to the given feedback, the Waterfall method could have delayed the need for change until after all required steps had been completed and the product was cleared for testing. It is therefore possible to argue for the method's positive impact on the progression of the project, as it ended up handling the various feedbacks effectively and consecutively.

The effectiveness of using Scrum can also pose as a second argument in the method's positive impact on handling the incoming feedback. As stated in the attached minutes of meetings for meeting five, the team chose to switch from programming the application's GUI in JavaFX to using the Scene Builder tool in the tenth week into the project. This decision was based on feedback from the student assistant, due to the effective nature of the tool. On the one hand, this switch contradicted with the team's initial vision to use this project to gain experience with JavaFX, which made itself relevant in a parallel project in IDATT2001 programming 2. On the other hand, the switch can also be viewed in the light of Risk A outlined in the vision document's risk analysis. The limited time allocated for development was regarded as a major risk as it could potentially lead to the final product not meeting the requirements for the project. As a result of this conflicting argumentation, the team chose to follow the advice after discussing the conflict in internal Scrum meetings. The choice can largely be justified on the basis of the desire to deliver within the final deadline April 28. This has been a consistent vision for the team, as communicated in the Gantt chart, the Vision Document's result goals and several places in the Collaboration Agreement. This example once again illustrates how Scrum facilitated responding to ongoing feedback that contradicted the team's predetermined visions.

Finally, it would be appropriate to highlight the team's handling of the conflicting feedback given in the user tests. As noted in Chapter 4 Results, different opinions towards the appearance of the application were for instance expressed between the client and the remaining test users. While the client wished for keeping the layout from the Wireframes, other test users pointed out a desire for a more coherent appearance of the application. This provoked a conflict of interest within the team. On the one hand, this can be justified by the fact that the client had been prioritized throughout the project. Inputs from the client therefore weighed heavily on the choices made in the development of the application, as the final product was constructed with the intention that the client would use it himself. On the other hand, the input from the other test users appealed to Jakob Nielsen's *10 Usability Heuristics*, which expressed the positive consequences of a consistent and familiar design. The conflict therefore revolved around whether going against the client's wishes or the other inputs who find support in interaction design theory. The team considered the pros and cons of both solutions. In the final sprint the budget overview and dashboard underwent a revision, with the argument of familiarity as decisive for the choice made. The thesis that users do not always know exactly what they want can also be used as a defence for this reasoning. Demonstrably, this was also realized in this project, as the client expressed satisfaction with the final product and had nothing to delay the decision made by the team regarding the design layout.

Universal Design

During the development of the application, a discussion arose about the choice of colours for the pie chart on the budget page. On one hand, it was suggested to use red and green colours, as they are typically associated with expenses and income. On the other hand, it was argued that using more accessible colours would enable a wider range of users to access the charts. Red-green colour blindness is the most common form of colour blindness, and using red and green would make it difficult for people with this condition to distinguish between the colours. Furthermore, cultural interpretations of colours can vary, and not all users may associate red with expenses and green with incomes. Nonetheless, it was mentioned that users with red-green colour blindness could enable the application’s colourblind mode to differentiate between colours easily. It was also mentioned that the client said he liked the green and red in the charts in the final test round. To ensure the application’s compliance with WCAG 2.1 principle 1 - Perceivable, the question was brought up at meeting seven. Student Assistant Daniel Andre Evensen confirmed the application would still be designed according to this principle, even with green and red charts, as the application has a colourblind mode. To reach a final decision, a user test was conducted with a person who has partial red-green colour blindness. The tester was able to distinguish between the green and red bars, lines, and sectors on the charts, as mentioned in Chapter 4. In the end, it was decided to use red and green colours for the charts, with the assurance that users with red-green colour blindness could easily switch on the application’s colourblind mode to differentiate between the colours.

Although the application is designed in accordance with the principles from WCAG, there is still room for improvement in terms of accessibility. The WCAG 2.1 has three levels of success criteria, and to meet the requirements set in the vision document, only the lowest level of conformance was necessary. However, achieving a higher level of conformance could have made the application more user-friendly. Prioritizing conformance could have resulted in a higher risk of incomplete or poorly executed features due to lack of time, as identified in the vision document’s risk analysis, Risk B. When selecting the product features, it was therefore important to carefully consider the extensive list of options. In the process, the team prioritized features such as a budget overview, accounting overview, recurring payment, visualized data, and a dashboard page. These features were prioritized above a higher level of conformance, as they were specifically requested by the client and deemed to be more important for making the application more useful to its intended audience. Therefore, it was agreed that meeting the lowest level of conformance would suffice, with any remaining time being spent on improving conformance.

To better comply with Success Criterion 1.4.1 - Use of Color, the team attempted to use patterns instead of colours in the charts in the application. Additionally, to meet Success Criterion 1.3.3 - Sensory Characteristics, more fitting icons could have been placed next to text, for example a plus sign for "Add new income" and a trash bin to identify the “Delete” button. Unfortunately, limited development time, Risk A, caused these details to be left out.

Chapter 6 Conclusion

As far as the team is concerned, the final version of the program turned out as a success. The comprehensive goals were met, and the program is fully functioning in its current state. To be sure, there are many areas in which the application could be further refined to make the interaction and workflow feel smoother and more efficient, but considering it being a first release version, it performs according to the project goals. Business-critical features have been implemented and improved throughout the project. Of the planned release features that were dropped, none were of a character that would significantly impact the performance or usability of the product, and the team do not consider their absence to be of significance.

This shows that the team was not too ambitious in its initial vision and goals and kept a sober focus on providing core functionality rather than excess features. While it can be argued that a somewhat conservative approach may have constrained the vision and limited willingness to experiment, it has nevertheless resulted in the core functionality being given due attention. Since the main goal of the project was to provide a functional application this focus seems well placed.

As for team and personal development, it was made explicit in the Vision Document that one of the main goals for the project was to provide educational value. During the project, the team showed a willingness to change course according to feedback from both internal and external sources, such as making changes to the design and realizing that daily standups had been neglected. The project has required all team members to take an active part in the process of team collaboration and project management, as well as the programming of a complex piece of software. This, combined with having to consider the intricate relationships between code, interaction design and regulatory requirements, have provided all members with valuable experience, which in turn marks yet another goal as achieved.

Regarding feedback from the client, student assistant and course lecturer, as well as participants in the final user tests, the overall impression was positive. Despite the inefficiencies and areas of improvement that have been pointed out, both client and test participants could certainly see themselves using the program, given assurances of further updates and polish.

Further work

Team believes there are certain recommendations that could further improve the program and its user experience. Firstly, the program currently supports only a limited range of year selections, from 2020 to 2030, and expanding this range would enable users to log finances further back and plan further into the future. Since the current client intended to use the program for private purposes, this was not seen as a critical issue in the first iteration of the software. Secondly, the team recommends adding a Dark mode feature to enhance the user experience by reducing eye strain, as noted in the project's vision document. Additionally, the currency selector currently only changes the name of the currency used in the program. It would be beneficial to improve this feature by adding backend logic that can convert the amounts displayed to the selected currency. Currently, users must choose one form of currency and stick with it to avoid confusion when adding entries.

In addition to improving the calendar's range, the team strongly recommends enhancing the recurring entry system. Developing this feature proved to be a more significant undertaking than initially believed. The current system recreates the entry object for a certain number of times when the repeatable checkbox is selected, but it complicates editing an entry, as the user must edit each instance individually. Although the team recognizes this is not an ideal solution, it was considered adequate given the time constraints. The team also recommends allowing users to choose the number and units for repeatable entries interchangeably and implementing a login feature with credentials stored in a database, simplifying datafile storage. These improvements would require substantial development but are considered important recommendations.

Throughout the project, the team prioritized adapting the program to the client's preferences. The client was pleased with the final result and expressed interest in continuing to use the program. However, during the user test, the client had a few requests for improvements to be considered for the next iteration. He also wanted a "+" button added to the "Add income" and "Add expense" sections on the Accounting page, for adding a new category. Additionally, the client found the summarization on the Dashboard's bottom bar confusing, and requested a reconfiguration of that function without specifying how. Lastly, the client requested the Dashboard's left chart to display a monthly trend instead of the current yearly trend. Other than these suggestions, the client displayed enthusiasm when interacting with the financial program and looked forward to using it in the future.

Repository

GitLab Wiki: https://gitlab.stud.idi.ntnu.no/team_01-idatt1002/project-assignment-idatt1002-y2023_spring-t01/-/wikis/home

GitLab issueboard: https://gitlab.stud.idi.ntnu.no/team_01-idatt1002/project-assignment-idatt1002-y2023_spring-t01/-/boards/6551

GitLab JavaDoc pages: https://team_01-idatt1002.pages.stud.idi.ntnu.no/project-assignment-idatt1002-y2023_spring-t01/

References

Beck, K., et al. (2001). *The Agile Manifesto*. Agile Alliance. Accessed 09.04.2023 from: https://agilemanifesto.org/

Cohen D., Lindvall M. & Costa P. (2004). *An Introduction to Agile Methods*. Fraunhofer Center for Experimental Software Engineering. Accessed from: http://robertfeldt.net/courses/agile/cohen_2004_intro_to_agile_methods.pdf

Datatilsynet (2018). *Dine rettigheter*. Accessed 10.04.2023 from: https://www.datatilsynet.no/rettigheter-og-plikter/den-registrertes-rettigheter/

Digitaliseringsdirektoratet (2023a). *EUs webdirektiv (WAD)*. Accessed 07.04.2023 from: https://www.uutilsynet.no/webdirektivet-wad/eus-webdirektiv-wad/265.

Digitaliseringsdirektoratet (2023b). *WCAG-standarden.* Accessed 07.04.2023 from: https://www.uutilsynet.no/wcag-standarden/wcag-standarden/86.

Eggen, N. A. (1999). *Godfoten*. Oslo: Aschehoug.

Forskrift om universell utforming av IKT-løsninger (2023). Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger (FOR-2013-06-21-732). Accessed 07.04.2023 from: https://lovdata.no/forskrift/2013-06-21-732/§4.

Kathayat, S. B. (2023). *Software Testing*. IDATT1002 lecture slides.

Moelk, B. (2018). Users don’t know what they want (and that’s ok). *Medium*. Accessed 07.04.2023 from: https://medium.com/brainendeavor/users-dont-know-what-they-want-and-that-s-ok-ef7d55d0e8e.

Nielsen, J. (2001). *First Rule of Usability? Don’t Liste to Users*. Accessed 07.04.2023 from: https://www.nngroup.com/articles/first-rule-of-usability-dont-listen-to-users/.

Nielsen, J. (2020). 10 Usability Heuristics for User Interface Design. *Nielsen Norman Group*. Accessed 07.02.2023 from: https://www.nngroup.com/articles/ten-usability-heuristics/.

NITO (2021). Etikk – for ingeniører og teknologer. Oslo: NITO. Accessed from: https://www.nito.no/contentassets/db24bd06c0ec446384162ad5b2229ff8/nito-etikk-for-ingeniorer-og-teknologer-2021-hefte.pdf

Norman, D. A. (1983). Design principles for human-computer interfaces*. ACM Digital Library*. Accessed 07.04.2023 from: https://dl.acm.org/doi/abs/10.1145/800045.801571.

Personopplysningsloven (2022). Lov om behandling av personopplysninger (personopplysningsloven) (LOV-2018-06.15-38). Accessed 10.04.2023 from: https://lovdata.no/dokument/NL/lov/2018-06-15-38/KAPITTEL_gdpr#KAPITTEL_gdpr

Sandstrak, G. (2023). Profesjonsetikk. Idatt1002 lecture slides.

Tesdal, N. (2020a) *UML*. NTNU. Institutt for datateknologi og informatikk.

Tesdal, N. (2020b) *Systemutviklingsmetodikk*. NTNU. Institutt for datateknologi og informatikk.

U. S. General Services Administration (2023). *Wireframing.* Accessed 08.04.2023 from: https://www.usability.gov/how-to-and-tools/methods/wireframing.html.