Personal Budgeting and Banking Program

Project Main Report

Version 1.3

Revision History

Date	Version	Description	Author
12.04.2023	1.1	First revision	Ingar S. Asheim, Lars M. Nilsen
19.04.2023	1.2	Second revision	Ingar S. Asheim
27.04.2023	1.3	Final revision	Ingar S. Asheim, Lars M. Nilsen, Trygve Jørgensen

Abstract

The compulsory project in IDATT1002 involved making a fiscal management application for a customer chosen by the team. This report showcases the results of this project and provides a discussion of the theory, methods, and outcomes for Group 21's delivery of the mandatory project in IDATT1002. Our task was to create an application for a private individual or small business to help them get a better overview of their income and expenses, aid in budgeting, and improve their overall economy. The development process followed several criteria and deadlines while documenting the entire process to be included in the main report.

Preface

IDATT1002 is a subject that aims to comprehensively cover the various aspects of software and product development. This includes defining requirements from the customer, creating the product vision and other design documents, setting up a DevOps environment, programming, doing user tests, and writing documentation. The purpose of this project is to teach these elements through practice.

This assignment was mandatory and accounts for 45% of the course grade. The initial task involved identifying a customer for whom the application would be developed, and in this case, it was a friend of a team member. We conducted interviews to gather the product requirements and subsequently formulated a vision document based on the collected information, created necessary diagrams, and set up a DevOps environment to aid in development.

Throughout the development process several methods were utilized to reflect about the state project. We employed the Scrum methodology to efficiently communicate, and delegate work where needed. We refined our approach based on feedback received during meetings with student assistants and instructors and employed user testing with the customer at various stages of development.

During this semester, we have had valuable learning experiences. We have both efficiently and safely learned to use Git in a team. We have learned about new design patterns and structures that were considered when designing the application and we got to employ the Scrum methodology for the first time.

The team would like to thank the instructors, Grethe Sandstrak, Muhammad Ali Norozi and Surya Kathayat, the student assistants, especially our assigned student assistant Nils William Ormestad Lie, for their feedback and guidance. We would also thank the customer Tormod Malmin for his active participation in the project.

28.04.2023 NTNU Gløshaugen

Ingar Solveigson Asheim 

Trygve Jørgensen 


Ari Maman

Lars Mikkel Lødeng Nilsen 

Table of contents

Assignment description

Our task was to create an application for a private individual or a small business by helping them get a better overview of their income and expenses, thereby improving the user's economy. The application should aid in budgeting as well as accounting, and we ourselves were responsible for finding a person or a small business that could act as our customer. The whole development process had to follow several criteria and hit several deadlines to be accepted as a fulfilled assignment, all while documenting the entire process to be included in the main report.

Our client was Tormod Malmin, age 19. He works a part-time job and does freelancing in his freetime by creating furniture from recycled wood. He had a experienced a need for a way to plan and track his freelancing project, which became the core feature of the product we developed.

Tormod Malmin, 19, Tlf: 928 62 003, Mail: Tormod.malmin@kleppnett.no

1. Introduction

This section serves as the introduction for this report. This is the report of group 21. This report has revision history, abstract and preface before the contents of the report. Further is the main content of the report structured with a numbered system, with an overview in the table of contents. Lastly there are links to the repository of the project, references and an appendix.

1.1 Definitions, Acronyms, and Abbreviations

CI/CD: Continuous Integration/Continuous Development
DevOps: Concatenation of Development and Operations
IDE: Integrated Development Environment
LLM: Large Language Model
MVP: Minimum Viable Product
OOAD: Object-oriented analysis and design
OOP: Object-oriented programming
UML: Unified Modeling Language
UX: User experience
GUI: Graphical User Interface

1.2 Glossary

IDE: A piece of software that provides both a text editor and different tools to aid with programming.
Library: Pre-written code, enabling users to avoid rewriting solutions for a problem, thereby saving time and resources.
Unit test: A piece of code that tests a small part of the code in an application.

2. Theory and relevant literature

2.1 Prior knowledge and experience

All students in IDATT1002 have completed the same courses prior to undertaking this project. Two of them are specifically relevant: "IDATT1001 Programming 1" and "INGT1001 Introduction to the Engineering Profession". Alongside this project, we are also enrolled in "IDATT2001 Programming 2", which serves as a continuation of Programming 1.

The INGT1001 course involved a collaborative project that shares similarities with this project, including project management, collaborative programming, and documentation. The knowledge acquired from this course is highly relevant and directly applicable to this assignment.

2.2 Programming design principles, methods and patterns

2.2.1 Object oriented programming

OOP is a programming pattern that organizes data and code objects (Barnes & Kölling, 2017). Each object is based on a class that sets the structure for what variables an object can store and what methods it can run (Barnes & Kölling, 2017). This allows for proper structuring of a program's functionality by breaking each function into smaller pieces. OOP also improves modularity and reusability as each class can be instanced to the necessary object where needed (Barnes & Kölling, 2017).

2.2.2 Serialization and deserialization

The Java Virtual Machine will store all data for a Java application in the heap which itself is stored in the RAM. This means that when closing the application, all data from the application will be cleared and permanently lost. As in most applications it is necessary that the user to store information after closing the program, writing data to a separate file stored on the system drive is how almost all modern program's function. Since saving data to a file can be done in numerous different filetypes with many different formats, serialization of class instances offers a reliable and easy way store information.

When serializing an object, all the object information, both methods and variable data, is converted to a single byte-stream that can be written to any file. The byte-stream is the exact binary representation of how the Java language stores the object in memory during runtime and is therefore unreadable to humans. This means getting the stored object the serialized file is achieved through a inverse deserializing process.

2.2.3 Singleton design pattern

When running an application in the Java Virtual Machine, every class can have multiple instances or objects created from it. Each object can vary in the information they store, but in the case that an object transfers some

information to another, it must be done through a method. This is because every object variable is constrained to the object itself. Instead, passing data from object A into object B can be achieved using the singleton design pattern.

A singleton class can be designed to store data that should be available to any class at any time in the entire application (Wikipedia contributors, 2023f). Such a class must have a private constructor to avoid any other class creating an instance from it (GeeksforGeeks, 2023). This class will have a static class variable of itself that will function as the only instance of this class that can ever exist in the applications runtime (GeeksforGeeks, 2023). Lastly, a static method assigned to get the singleton instance must be public. When called it will check if there already exists an instance of this class, if not it creates a new instance for the first- and only-time during runtime. With such a class, any other class can call the static singleton method and receive the singleton instance, retrieving and updating its information.

2.2.4 Model View Controller

A complete application with a graphical user interface, backend logic and file storing can quickly result in a high complexity project with lack of modularity and structure. If the application to be developed contains a GUI, following the model view controller architecture will clearly structure the project and allow for proper expanding of application functionality.

In a Java program, the model, view, and controller work together to create an application (*Java SE Application Design With MVC*, n.d.). The model stores and manipulates program data, while the view displays information to the user and receives input (*Java SE Application Design With MVC*, n.d.). Java applications often use .fxml files for this purpose. The controller's job is to pass information from the view to the model while updating the view with any changes from the model. It does not store information itself.

2.3 Product design

2.3.1 System modeling, analysis, and development

Object-oriented analysis and design, or OOAD (Object Oriented Analysis and Design), is a technical approach that applies object-oriented programming principles to product design and problem solving (Wikipedia contributors, 2023b). It can be used for several tasks such as the development of an application, system, or business (Wikipedia contributors, 2023b). It fits well for software development as it uses OOP principles. UML models are frequently used in the development process to clearly plan out system structure and flow, which allows for communicating the product to non-technical stakeholders (Wikipedia contributors, 2023b).

2.4 User experience design

User experience design (UX design) is a term that refers to all aspects of a user’s experience when using a product. This includes ease of use, utility, and efficiency of the product, as well as any other processes needed to use the product.

2.4.1 Don Normans design principles

Don Normans design principles are general guidelines that can be applied to all products where the user experience is a top priority (Dahl, 2017).

Affordance is the concept of an object inherently possessing the qualities necessary to infer how it can be used. The user therefore should immediately have an intuitive understanding of the objects functionality upon perceiving it. **Consistency** should be upheld throughout the whole design, as it minimizes confusion for the end user. The designer should **constrain** the user to only be operating within certain boundaries, thereby reducing the potential for user errors. Whenever the user does an action, the system should give **feedback** to signal that the users action was detected. **Mapping** is the correspondence between the controls and their respective functions, proper mapping means intuitive understanding of function. **Visibility** emphasizes the importance of making parts of the design that users will interact with easily visible.

2.4.2 Wireframes

To aid in the design of a GUI (graphical user interface), wireframes are a tool that is commonly used. Wireframes are a low-granularity visual representation of the layout and structure of a user interface. They are intended to be created and updated quickly based on feedback from user tests. Wireframe tools allow for the wireframe designs to closely resemble the final GUI, and even let the wireframes interact with inputs for more realistic user tests. This provides a way to ensure that a solid design is achieved before production of the application begins, safeguarding against costly redesigns during development (Gupta, S. 2023) (Perera. R. 2023).

2.4.3 Usability tests

Usability testing refers to the practice of having users not involved in the development of a product to test said product. The goal is that the user to use the product in a way that seems natural, every hiccup the user experience then becomes a data point to areas that can be improved. These tests are conducted by letting the user interact with the product and comment on their experience. The developers focus can differ from the customers focus, and a usability test helps align in what should be improved upon (Usability testing, 2023).

2.4.4 Universal design

Universal design is the process of creating a design that is accessible to as many people as possible. The Web Content Accessibility Guidelines (WCAG) 2.1 covers a wide range of recommendations for making Web content more accessible. While the guidelines are made with webpages in mind, they are also applicable for other digital applications. Following these guidelines will aid in creating a design that accommodates the needs of people with a wide range of disabilities but will not address every user need for people with these disabilities. (MDN, 2023).

The first principle in WCAG 2.1 is titled “Perceivable” and sets criteria relating to content that the user can sense in any way, shape or form. This implies providing robust alternatives to the way the user perceives content on the webpage, including proper use of colors, proper resizing, options when consuming time-based

media etc. (MDN, 2023)

2.5 Software development and tools

2.5.1 Agile methods and Scrum

Scrum is an Agile framework that consists of a set of values, principles, practices, and artifacts that help a team cooperate successfully. There are three main roles in the Scrum framework: the product owner, the scrum master and the development team. (“Fundamentals of the Scrum Methodology | Blog Wrike”)
The development process is split into sprints, with each sprint consisting of several steps. Before the sprints start, the requirements for the product are split into goals and the tasks necessary to complete those goals. These goals become what is called the sprint backlog.

The team initiates the sprint by starting the process of resolving and achieving goals in the sprint backlog. In parallel, the Scrum Master maintains a list of impediments that are preventing the team from achieving their goals. Daily Scrum meetings, called stand-ups, are held where the team discuss the progress they have made, what they are planning to do next, and problems they have encountered. At the end of a sprint, the team holds a sprint review and a sprint retrospective. The sprint review consists of the team documenting the work that was done during the sprint. In the retrospective, they discuss how the development process itself went and how it could be improved for the next sprint (Scrum.org, 2023) (Wikipedia, 2023e).

2.5.2 Development tools

2.5.2.1 Version control and DevOps platforms

Version control is the practice of tracking and managing changes in code. (Atlassian, 2023) This is performed by software that is specialized for this task. When changes in the code are made, the developer can use the version control software to take a snapshot of the files at that point in time. It is then possible to go back to or later versions with the new changes. It also collaborating between developer close to frictionless, as they can make changes to the same files without interfering with the other’s changes (RhodeCode, 2023).

These snapshots are often stored remotely on a server, and DevOps platforms are commonly used for this task as well as integrating other DevOps features. CI/CD functionality is commonly included, which verifies that a new snapshot which is pushed to the server passes all included tests that, and then deploys the updated application to the production server. This enhances the effect of the iterative development cycle, as users can give feedback and find bugs for slight changes, which will be easier to fix.

IDEs are the very tools that are utilized when developing software. In addition to being a text editor where the programmer writes their code, IDEs provide a multitude of tools that aid in programming. Some of the most important functions that ship with every IDE are the debugger, syntax and check style warnings. This shields the developer from making application breaking error that usually propagate from a single misplaced or missing symbol. If the application does not function as expected the debugger will aid in analyzing the program state at any point in its execution. Recently AI assistant have become available that assist the user in filling out

boilerplate code and hard to remember syntax, boosting the users productivity.

2.6 Documentation

2.6.1 Reporting

Documentation is an important part of development. The vision document serves as a comparison when authoring the final report, as it contains the project goals that might or might not have been fulfilled. The final report communicates the results of the project to stakeholders, reflection over the developmental process to the developers, and generally as a document anyone can look to for reference in the future.

2.6.2 Javadoc

A very critical category of documentation is covering all code functionality. Java's built-in solution is Javadoc that uses a standardized commenting format above every class and method. Each comment allows the developer to quickly get an overview over the class or method function, with the possibility of hyperlinks to other documented instances. There are tools that can parse the Javadoc comments and generate formatted versions that can be deployed to webpages. This is useful for future developers, as it can make the process of understanding the existing code easier.

2.6.3 Wiki

In modern software development, encyclopedias in the form of project wikis is a standard practice. A wiki is a repository all project documentation, in addition to all other information that is relevant to the given project. This should therefore include diagrams and models, user manuals, system requirements, project structure, etc. (Merriam-Webster, 2023).

3. Method

The reasoning behind most of the choices regarding to methodology, development standards and technologies are that they were obligatory in completing the assignment. There were also several requirements on the project reporting itself, however, it's still useful to reflect on why the assignment chose these requirements in the first place.

3.1 Programming design principles, methods and patterns

As OOP is how java is designed to be programmed, all best practices related to OOP was followed to the best degree. The single responsibility principle was the framework for how each class in the project model should functioned, resulting in a very vertical class structure. It starts at the bottom with a Transaction interfaced class stored in a Bookkeeping interfaced class that is stored in either a Project or a MonthlyOverview class that again is store in a Registry class which finally is stored in a User singleton class. Here each class responsibility area is completely contained within its own class.

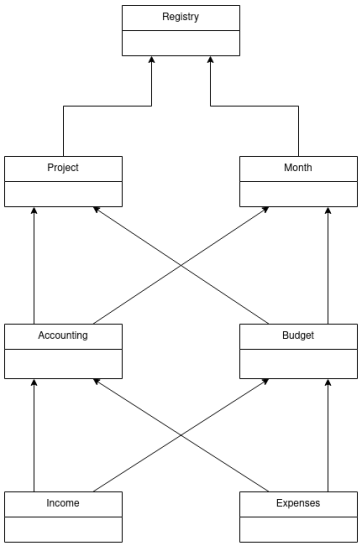


Figure 1 Simplified domain model

3.2 User experience design

Although the following general design principles should be taken into account from the very start, prototyping of the GUI was initialized immediately as the project requirements were set. The very first draft of the wireframes were therefore designed with little to no guidelines, other than what seemed like intuitive and common-sense design. This resulted in a good base for how the program should look and function, and the design was immediately refactored fulfill the criteria of the given design principles.

3.2.1 Don Normans design principles

As many of Don Normans design principles could be categorized as common sense,
most guidelines were already fulfilled after the first wireframe draft, although some parts of the GUI required rethinking.

Affordance related to the visuals of buttons, input fields, etc. As these where presets from the JavaFX library were every interactive element already designed to be as understandable and readable as possible. Curtain unwanted behavior like clicking inaccuracies would propagate from non-standard elements like icon buttons and had to be refactored. **Consistency** was considered by formatting all the pages have the same width and height, and buttons and elements with similar functions would be sized and styled the identically. Page layouts were also formatted to have similar functions in similar locations. Most **constraints** were put in place later in the development process as the project model began to react on invalid processes. Certain button would be disabled when their illegal to press, and invalid inputs would be stopped given a warning message (Dahl, 2017)

Feedback given to the user is only visual as any auditorial feedback was not deemed necessary for the

application. Feedback is mostly shown as updates to tables when new information has been added, but the user will also receive popups asking them to confirm if they want to delete or cancel a project. The **Mapping** of every element was considered in context to the western reading order, which goes from top-left to bottom-right. Title and static menu buttons are placed at the top, so it's used as a fixed reference point in the entire application. Following the reading order, interactive elements are placed to the left, with purely informational elements to the right. Finally, finalizing processes like saving or deleting are placed in the lower right corner as it's the last bit of information the user should perceive. The **visibility** of any information at any moment is mainly dependent on which mode the user has selected, and if a warning is necessary to display (Dahl, 2017).

3.2.2 Universal design

Same as with the Don Normans design principles, many of the criteria in the WCAG 2.1 perceivable principles were already accounted for as the GUI was designed with clarity and common sense in mind.

The criteria **"**Sensory Characteristics" specifies**,** "...content does not rely solely on sensory characteristics of components such as shape, color, size, visual location, orientation, or sound." All interactive elements like buttons or fields have text explain them or at the very least universally understood symbols like "+" and "x" in the case adding and deleting. The "Orientation" criteria were intentionally not fulfilled as the application was only designed for PC. Compatibility with portrait mode was not developed as only a miniscule amount commercial laptops allows for such functionality (MDN, 2023).

The "Use of Color" criteria states that "Color is not used as the only visual means of conveying information" and is taken care of as everywhere color does convey information, there is text to further specify its meaning. The only exception is in the monthly overview where every income is marked with green and expenses with red, as it was assumed that this would be intuitively understood. The "Contrast" criteria was confirmed with a contrast calculator. As is, all instances of text on background color have a contrast ratio larger than the minimum 4.5:1. Lastly, the "Reflow" criteria was modified to a more achievable goal of 720px by 1280px as implementing such a dynamic page reflow in JavaFX required too much refactoring too late in the project. This decision was also taken as it is the minimum resolution on modern laptop screens that are sold today (MDN, 2023).

3.2.3 Usability tests

The user tests were conducted according to the section on usability test in 2.4.3. The customer was asked to test the program throughout the process of the development of the product. Because of the distance between developers and customer, the testing was conducted through the application of discord, which allows for calls and for screen sharing. This allowed for a showcase of the product. The tests were done in person when the customer contact was in the same county as the customer. Reports were made of each test and gave the developers a good dataset from which the product could be improved upon.

3.3 Development

In this project we used the Scrum development methodology, the Java programming language, the JavaFX GUI framework and using Git with GitLab as the DevOps platform were mandatory to use for the assignment. The Scrum methodology is an industry standard for software development and is therefore highly relevant to learn for computer engineering students.

The Java programming language, while slowly falling out of favor for newer alternatives, is still a useful language to learn because of the considerable number of products and systems that are written in it. It is also a suitable choice for beginner programmers due to the amount of documentation, guides, and other resources to learn and get help with the language that are available. It is also an exclusively object-oriented language.

As mentioned in 2.1, the subjects Programming 1 and 2 both use Java, and Programming 2 also uses JavaFX and Git with GitLab for DevOps. Using these technologies in both courses provides more practice and means the students do not have to learn more. Both the increased practice and not having to learn different technologies improve the learning experience.

Commented [TJ1]: Kan komprimeres?

3.3.1 Technologies

The three main pieces of software that were used during development were the IDEs and Git. One of the developers used VSCode for his IDE, while the others used IntelliJ. Both are IDEs that are widely used in industry, and it is therefore highly relevant to learn how to use them. Git is the most popular version control system, and GitLab is centered around the use of Git. Using Git was therefore necessary and is also a good choice for its relevancy for later use in the team members' careers (Atlassian, 2023), (RhodeCode, 2023).

The hardware used was the team members' computers and, by proxy of using GitLab, their servers too. We experienced a small number of technical problems while using GitLab. Setting up the script for the CI/CD pipeline proved challenging but was eventually set up successfully with aid from Kathayat.

3.4 Documentation

3.4.1 JavaDoc

When we wrote the JavaDoc we tried to follow the industry standard. This meant that the javaDoc should cover each class, method and variable. We used @param for parameters and @throws for exceptions thrown in all methods and constructors. We also used @return for methods that returnes a value.

3.4.2 Wiki

We made multiple diagrams and models for the wiki, including a domain model, class diagram, sequence diagram and use case diagram. The purpose of the diagram is to make the program easier to understand. We

also wrote an installation manual that serves as help for the user to install the application, and a user manual that's going to help the user on how to use the application. When we wrote the installation and user manual, we tried to write it in a way that people with little coding/computer experience could understand it. We also wrote about more technical parts of the program such as persistence and project structure. The wiki also includes a link to the JavaDoc (Merriam-Webster, 2023).

3.5 Work and role distribution

This team consists of four individuals, each with unique roles and responsibilities to contribute to the team. This is to ensure that everyone gets to show off their skill set and become irreplaceable assets to the team. The four team members and their respective roles are:

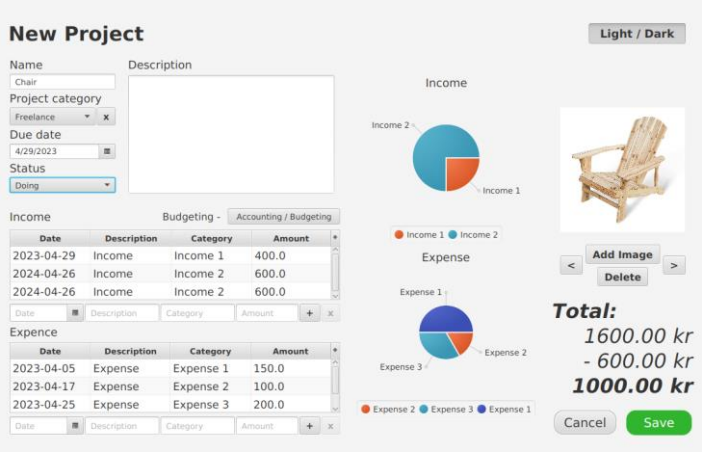
Name	Description	Responsibilities
Trygve Jørgensen	Developer & Team leader	In addition to developing the program, Jørgensen will be responsible for keeping the team's progress on track and leading meetings, both during and outside of Scrum.
Ingar Asheim	Developer & Documentation manager	As an experienced programmer, Asheim shall provide guidance during the programming phase of development.
Lars Mikkel Nilsen	Developer & Meeting organizer	Lars will be responsible for arranging meetings, writing documentation and helping develop the program.
Ari Maman	Developer, referent & customer stakeholder representative	Maman is the stakeholder representing the customer and shall communicate the needs of the customer to the development team and vice versa.

4. Results

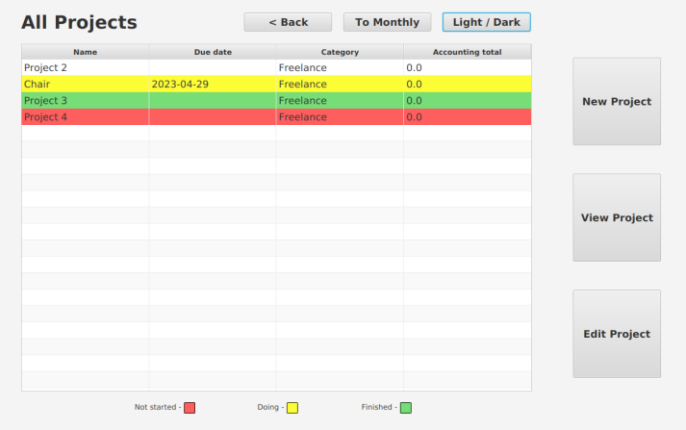
4.1 Finalized product

In section 2 of the vision document, the project goals are described. They are split into three types: Impact, result, and process goals. It is impossible to assert the success of the impact goals, as they can only be assessed after the customer has received and used the application for a period of time. However, all results and process goals were achieved. All stakeholder/user needs in listed in the table in section 3.5 of the vision document were also fulfilled, as were all items in section 4.2 in the vision document. Screenshots of the final application will be shown alongside an explanation of any goal(s) relevant to that part of the application. A

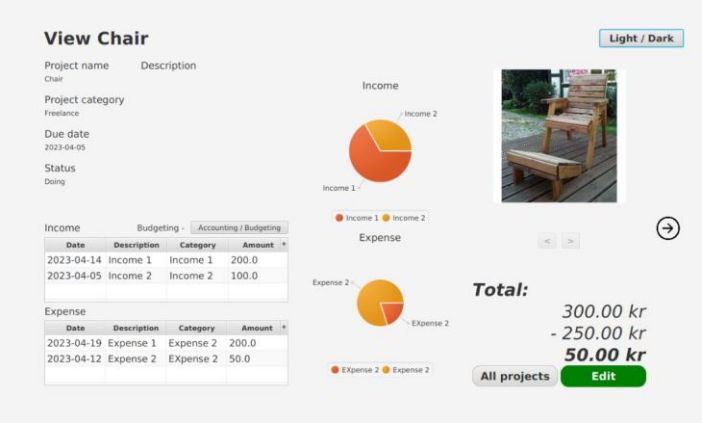
4.1.1 Projects page



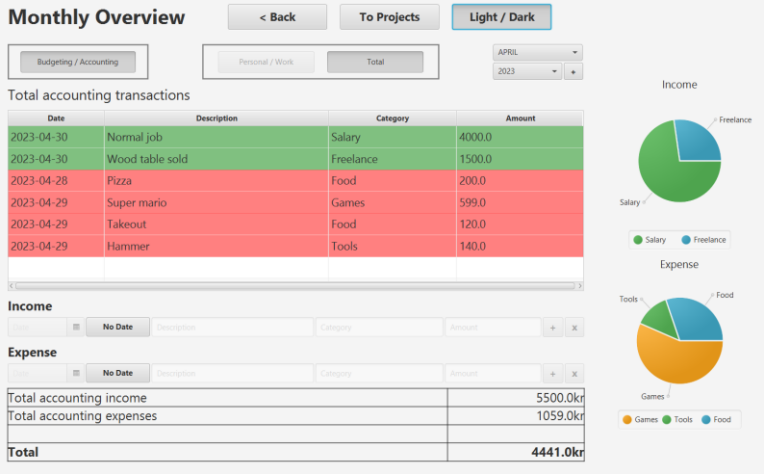
The screen for adding a new project. Addresses point 1 and 3 of the result goals and item 1 in the table in section 3.5 of the vision document.



This screen shows the overview of the different projects. Addresses point 1 and 3 of the result goals and item 2 in the table in section 3.5 of the vision document.



This shows the first project in the project view-mode, which is the implementation of the slideshow functionality mentioned as a proposed solution for item 2 in the table in section 3.5 of the vision document. Note the arrow on the right. Clicking this will bring the user to the overview of the next project. In addition to the aforementioned item, this also addresses point 1 and 3 of the result goals.



This shows the monthly overview page. These addresses point 1 and 2 in the result goals as well as the third item in stakeholder/user needs.

Commented [TJ2]: Er dette nødvendig, kan kuttes

4.2 Administrative results

At the start of the project, we created a GANTT diagram, containing what we thought at the time were all the internal and external deadlines. We divided the project into three iterations, since there were three mandatory

deadlines during the progress, therefore we also tried to make the diagram follow these three iterations. We added additional goals as the project progressed. We completed all goals before the external deadline, and most before the internal deadline. In the image under you can see all the goals on the GANTT diagram, with a start date, finish date and the goal deadlin.

Tasks	Start	Finish	Deadlines
Set application requirements with customer	09.01.2023	06.02.2023	
Create and sign employment contract	09.01.2023	06.02.2023	10.02.2023
Set up a new project in GitLab	09.01.2023	06.02.2023	
Draw Wireframe Prototypes	06.02.2023	26.02.2023	03.03.2023
Write Vision Document	06.02.2023	26.02.2023	03.03.2023
Draw Domain-model	06.02.2023	26.02.2023	03.03.2023
Code Minimal Viable Product (MVP)	06.02.2023	19.03.2023	24.03.2023
Prepare & Conduct User Test (MVP)	27.02.2023	19.03.2023	24.03.2023
Prepare & Conduct User Test (Application)	08.03.2023	10.04.2023	
Write Wiki	06.02.2023	24.04.2023	28.04.2023
Code Final Product	20.03.2023	24.04.2023	28.04.2023
Documentation	09.01.2023	24.04.2023	28.04.2023
Prepare and practice presentations	29.04.2023	30.04.2023	01.05.2023

Many of the goals mentioned had subgoals that were made into issues on the projects GitLab. We continually tried to resolve the issues to complete the main goals in the GANTT diagram. For example, the MVP had dozens of subgoals/issues that needed to be resolved before the MVP goal was completed.

One of the project requirements was that each team member used 120 hours +/- 10%, and as the timesheet indicates, every team member stayed in the 120 hours +/- 10% range. Based on the breakdown of activities in the timesheet, it is clear that most of our time went to coding, followed by documentation and team meetings. In the timesheet table sorted by activities, you can see that product development, which includes coding, has the most hours spent on it by far, followed by administration and documentation.

One of the first things we did in this project was to make a first version of the class diagram. The first version had all the same classes as we ended up with, but we have since then added new methods and variables to the classes.

5. Discussion

As described in section 4, the final application met all the criteria in the vision document. Based on the feedback from the customer during usability tests, it also fulfills his needs.

5.1 Scrum workflow issues

We were able to organize into a team with good working relations between the members, and work was

delegated appropriately. This resulted in each member being able to complete their allotted work within deadlines, ensuring a smooth development process. While the development process went well, it did not follow the Scrum methodology to a satisfactory degree, even though this was a central part to the assignment. This was in large part due to its mandatory status being discovered under the first meeting with our instructor.

Changing the established work process to the Scrum methodology proved to be difficult, and ultimately only parts of the process were properly fulfilled. This may be a result of poor efforts in restructuring the workflow, which could have been mitigated by a proper team meeting with focus on implementing the Scrum workflow. It is also important to note that since product development was never halted because of an incomplete scrum process, we did not see a need for a more engaging workflow process, and instead continued as usual.

5.2 End product success

As the final product achieved all vision goals, the strategy of function now, polish later was likely a big factor to delivering a robust product, rather than a visually appealing one with unwanted bugs. Although its unwanted behavior, the unintentional lack of Scrum process and documentation meant that more time could be delegated to product development, and possibly therefore resulting in a better product.

The product requirements that were formulated with the customer were from the very start a list of defined features. This meant that feature development was clear cut and undesirable phenomenon like feature creep rarely or never occurred. We were also early on advised by our student assistant to considerably lower the MVP goals, that proved to be vitally important in succeeding with the MVP, which in itself did not create unwanted roadblocks for developing the finalized product.

5.3 Technical product success

As soon as the product requirements were set, modeling of the application structure was designed to have a clear structure that followed best practice principles. Utilizing OOP design patterns such as interfaces proved itself to be extremely helpful won the line, as the number of methods needed to design were drastically cut and overall code readability and structure were improved. Employing functional programming such as lambda expressions and ternary operators allowed for smaller code footprint, drastically reducing the number of lines in the project. This also improved the debugging process as each process is contained in a smaller footprint.

Using git was also a continuous learning process where each commit would get better descriptions and branches would be more concise. We figured out early on that merge conflicts were quite common when older branches were merged into newer ones. We therefore adapted to a process where each old branch would be pulled to local, deleted from remote, rebased on top of master and then push again to so be merged into master. This show to be highly effective, resulting in minimal merge conflicts that saved us from much wasted time trying to sort out our Git repository.

6. Conclusion and further work

Based on the results presented in section 4 and the discussion held in section 5, we conclude that the project has been highly successful. The final product met all requirements and had all the functionality from the designs, with a high degree of focus on universal design, and makes this an accessible program for people with no knowledge of programming. We paid the utmost attention to the customer, but also to the student assistant and teachers' feedback, and used this to improve to product accordingly. We valued good coding practices and strived to improve our code in every way we could. While we failed to follow the Scrum methodology exactly, we still managed to work well as a team. In addition to creating a successful product, we learned a lot about it throughout the process.

An important key point in working with such products is the scope of the product. Narrowing down the eventual functions of the product is a smart way of conducting management. The bigger a project becomes the more unmanageable it becomes and can lead to burnout or alternatively misappropriation of effort. In narrowing down the project the team was able to manage effort in a meaningful way and eventually develop extra features that were discarded in the initial planning of the project.

Repository

Project Wiki: https://gitlab.stud.idi.ntnu.no/gruppe-21/idatt-1002-2023-21/-/wikis/home
Issue board: https://gitlab.stud.idi.ntnu.no/gruppe-21/idatt-1002-2023-21/-/boards/6913
GitLab pages for Javadoc: https://gruppe-21.pages.stud.idi.ntnu.no/idatt-1002-2023-21/apidocs/index.html
GitLab pages for JaCoCo: https://gruppe-21.pages.stud.idi.ntnu.no/idatt-1002-2023-21/

References

Atlassian. (n.d.-a). *What is Scrum?* [Video]. Atlassian. Retrieved April 24, 2023, from https://www.atlassian.com/agile/scrum

Atlassian. (n.d.-b). *What is version control | Atlassian Git Tutorial*. Retrieved April 24, 2023, from https://www.atlassian.com/git/tutorials/what-is-version-control

Barnes, D. J., & Kölling, M. (2017). *Objects First with Java: A Practical Introduction Using BlueJ*.

Definition of wiki. (2023). In *Merriam-Webster Dictionary*. Retrieved April 26, 2023, from https://www.merriam-webster.com/dictionary/wiki/

GeeksforGeeks. (2023). Java Singleton Class. *GeeksforGeeks*. https://www.geeksforgeeks.org/singleton-class-java/

Gupta, S. (2021, August 9). *How to Create a WireFrame: Step-by-Step Guide*. Springboard. Retrieved April 24, 2023, from https://www.springboard.com/blog/design/how-to-create-a-wireframe/

IBM Documentation. (n.d.). https://www.ibm.com/docs/en/elms/elm/7.0.1?topic=requirements-vision-document

Initiative, W. W. A. (n.d.). *WCAG 2.1 at a Glance*. Web Accessibility Initiative (WAI). Retrieved April 24, 2023, from https://www.w3.org/WAI/standards-guidelines/wcag/glance/

Java SE Application Design With MVC. (n.d.). https://www.oracle.com/technical-resources/articles/javase/application-design-with-mvc.html

Juks på eksamen - Kunnskapsbasen - NTNU. (n.d.). NTNU. Retrieved April 27, 2023, from https://i.ntnu.no/wiki/-/wiki/Norsk/Juks+p%C3%A5%20eksamen

Keepitusable. (2020, August 8). *What is User Testing? - Learn UX*. Learn UX. Retrieved April 24, 2023, from https://www.keepitusable.com/blog/what-is-user-testing/

Operable - Accessibility | MDN. (2023, April 12). Retrieved April 24, 2023, from https://developer.mozilla.org/en-US/docs/Web/Accessibility/Understanding_WCAG/Operable

Perceivable - Accessibility | MDN. (2023, April 12). Retrieved April 24, 2023, from https://developer.mozilla.org/en-US/docs/Web/Accessibility/Understanding_WCAG/Perceivable

Perera, R. (2020). What Is a Wireframe & How to Design Them: A Beginner's Guide. *Design Shack*. https://designshack.net/articles/graphics/what-is-a-wireframe/

RhodeCode › Version Control Systems Popularity in 2016. (n.d.-a). RhodeCode. https://rhodecode.com/insights/version-control-systems-2016#:~:text=Conclusions%201%20Git%20is%20the%20most%20popular%20version,have%20the%20%20smallest%20shares%20in%20this%20survey.

RhodeCode › Version Control Systems Popularity in 2016. (n.d.-b). RhodeCode. Retrieved April 26, 2023, from https://rhodecode.com/insights/version-control-systems-2016#

Robust - Accessibility | MDN. (2023, April 12). Retrieved April 24, 2023, from https://developer.mozilla.org/en-US/docs/Web/Accessibility/Understanding_WCAG/Robust

The Definition of User Experience (UX). (n.d.). Nielsen Norman Group. https://www.nngroup.com/articles/definition-user-experience/

Understandable - Accessibility | MDN. (2023, April 12). Retrieved April 24, 2023, from https://developer.mozilla.org/en-US/docs/Web/Accessibility/Understanding_WCAG/Understandable

Understanding WCAG 2.1 - Service Manual - GOV.UK. (2022, May 24). Retrieved April 24, 2023, from https://www.gov.uk/service-manual/helping-people-to-use-your-service/understanding-wcag

Updates to the Scrum Guide: The 5 Scrum values take center stage. (n.d.). Scrum.org. Retrieved April 24, 2023, from

https://www.scrum.org/resources/blog/5-scrum-values-take-center-stage

What is Universal Design? – The Universal Design Project. (n.d.). The Universal Design Project. Retrieved April 24, 2023, from https://universaldesign.org/definition

Wikipedia contributors. (2023a). Object-oriented analysis and design. *Wikipedia*. https://en.wikipedia.org/wiki/Object-oriented_analysis_and_design

Wikipedia contributors. (2023b). Version control. *Wikipedia*. https://en.wikipedia.org/wiki/Version_control

Wikipedia contributors. (2023c). Agile software development. *Wikipedia*. https://en.wikipedia.org/wiki/Agile_software_development

Wikipedia contributors. (2023d). Scrum (software development). *Wikipedia*. https://en.wikipedia.org/wiki/Scrum_(software_development)

Wikipedia contributors. (2023e). Singleton pattern. *Wikipedia*. https://en.wikipedia.org/wiki/Singleton_pattern

Yngve Dahl (2017). 30-Designprinsipper. Retrieved April, 2023 from:

https://folk.ntnu.no/baldurk/skolearbeid/MMI/Forelesninger%20MMI/30-Designprinsipper.pdf