## Team 1

# Baecon

# Vision

**Version <1.0>**

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 13/02/23 | 1.0 | First draft | John-Ivar Eriksen, Birthe Emilie Christiansen and Lotte Christine Walla Aune |
| | | | |
| | | | |
| | | | |

# Table of Contents

# Vision

## 1. Introduction

The overall objective of this vision document is to present the project assignment and its attached requirements, included in the course IDATT1002 Systemutvikling. This complex project is counted as a part of the exam in the course, with emphasis on working with external stakeholders and target users in a scrum-project. The aim of the project is to deliver a functional product, that serves as a solution that gives an overview of income and expenses for a user's personal finances. This includes both budgeting and accounting. Furthermore, the given task description also provides guidelines for identifying and engaging external persons as users and client. While the scope of the project on the one hand is limited by the task description, the client, on the other hand, will have the opportunity to add extensions based on their personal preferences and wishes. The purpose of this Vision document is hence to present the team's vision for the final product, bearing in mind the given scope and the remote client. The details of how the project fulfils these is further detailed in separate UML-models and other documents, such as the domain and use-case model and wireframe.

The name:

Baecon: "**B**ig B**ae**rum E**con**omy"

### 1.1 Definitions, Acronyms, and Abbreviations

| | | |
| --- | --- | --- |
| .jar | – | File ending, file format, for Java applications |
| F/LOSS | – | Free/Libre Open-Source Software |
| GUI | – | Graphical User Interface |
| JDK | – | Java Development Kit |
| JRE | – | Java Runtime Environment |
| JVM | – | Java Virtual Machine |
| LTS | – | Long Term Support |
| MVP | – | Minimum Viable Product |
| UI | – | User Interface |
| UML | – | Unified Modelling Language |
| UX | – | User Experience |
| WCAG | – | Web Content Accessibility Guidelines |
| Wireframe | – | Visual prototype of the graphical interface of the application |

### 1.2 References


Doran, G. T. (1981). There's a SMART Way to Write Management's Goals and Objectives. In Management Review. 70, 35-36. New York.

*Stene, M. (1999).* Vitenskapelig forfatterskap: hvordan lykkes med skriftlige studentoppgaver. *Kolle Forlag.*

### 1.3 Overview

This Vision document will serve as a presentation of the team's vision for the project leading up to a functioning economic application. The following section will consist of an account of the positioning of the project, which includes a description of the problem and product statement. Subsequently it will follow a concretization of the team's objectives, where the impact, result and process goals will be presented in respective order. A more thorough description of the stakeholders and users will then follow, where the emphasis will be placed on the roles of those involved in the project. Then it will follow an overview of the product. This section will among other things include an elaboration of details and a risk analysis commenting on the potential risks that could arise. This section is followed up by a review of application features, development constraints, quality range and prioritization of development features. This document is culminated by an account of relevant documents required by the project.

In this regard it will also be appropriate to account for the chosen content in the Vision Document that diverges from the given template. Due to the given framework in combination with the status of the project, as a student assignment, changes have been made. The omission is centered around the chapters relating to the costs of the project, as they are irrelevant to the development of the final product. This applies in particular to chapter 5.5, 5.7 and 5.8 described in the template. Chapter 10.3 Performance Requirements and 10.4 Environmental Requirements, from the template, have been deleted in favor of chapter 8 Quality Ranges and 5.3 Assumptions and Dependencies, respectively, where their details have already been covered. Chapter 12 Features Attributes is removed because it does not provide any other details of what is already accounted for in the other parts of the Vision document. It has therefore been considered redundant.

## 2. Positioning

This chapter provides a broad summary of the target audience of the product and the potential benefits it can bring to them.

### 2.1 Business Opportunity

Our team was tasked with developing a tool to provide private individuals or small businesses with an overview of their income and expenses, with the goal of improving their financial situation. For this project, we have chosen to develop a tool for a private individual.

### 2.2 Problem Statement

| | |
|---|---|
| The problem of | Efficiently tracking expenses and income |
| affects | Svein Magnus Walaas |
| the impact of which is | challenges in managing finances effectively |
| a successful solution would be | Accessible and easy to use<br>Viable for both budgeting and accounting<br>Contributing to improved finances |

### 2.3 Product Position Statement

| | |
|---|---|
| For | A private individual |
| Who | Wishes to improve their economy |
| Big Bærum Economy | Is a standalone Java application |
| That | Is custom made according to the client's preferences. The application aims to improve the user's economy by providing a comprehensive view of income and expenses through numerical data and visual representations. Furthermore, the application will be viable for both budgeting and accounting and give the user recommendations on how to effectively manage expenses. |
| Unlike | Other applications that can't be used for both budgeting and accounting and that often require a subscription to access all features. While spreadsheet software like Excel is a popular tool for managing finances, it can be challenging for inexperienced users to create an effective money management system in this software. |
| Our product | Is available free of charge and can be used for both budgeting and accounting purposes. It should be designed with user-friendly interfaces and clear visual representations of financial information. In addition, our product will have personalized features to fit the client's needs. |

## 3. Project Goals

Given the various stakeholders involved in this project, the team has created several goals

that will try to respond to the task description set by the course coordinators, while at the same time taking the client's wishes into account and assuring the team's own satisfaction with the final product. In order to achieve this composite undertaking, the need for comprehensible and well-defined objectives has arisen. Therefore, as a joint unit, the team has constructed objectives for the project following the guidelines of George Doran's SMART-goals template. That is, goals that are specified, measurable, accepted, realistic and limited by time and costs.[1] This allows for planning and structuring the work required in the progression of the project. Thus, the team can collectively maneuver to deliver a final product that meets the requirements set by the stakeholders. In his book on succeeding student assignments, Morten Stene has emphasized the importance of the goals being universally accepted internally for them to be considered as good.[2] The objectives of the team have been distinguished into three types: impact, result and process goals.

## 3.1 Impact Goals

The impact goal is directly linked to what the client's wants for the final product, as an application handling their personal finances. Therefore, the team's main impact goal for the project is to deliver a functioning economic application for the client to use on a day-to-day basis. By function as a comprehensible tool for handling budgeting and accounting, the team hopes that the final product strives to offer an overview of the user's personal finances. Consequently, the impact goal is to facilitate a simpler and structured everyday life for the end user. This can further be justified by the additional task requirement, demanding a feature advising for healthier economical choices.

## 3.2 Result Goals

The result goal of this project is directly related to the outcome of the final product. The goal is to deliver a fully functionally application by the 28th of April. As expressed in the collaboration agreement chapter 3.E, see attached documentation, the team has laid an emphasize on delivering the solution within the given deadline, and where all the team members are comfortable with both the content and quality. The result goal is also related to the client. The aim is for the product compete with alternative solutions and function as an attractive replacement. In order to achieve this, it will be appropriate to consider the client's input and wishes in the development of the final product. Despite this, result is also regulated by the requirements of the assignment, which states that the solution must be designed according to the WCAG 2.1 principle 1 Perceivable. This provides natural consequences for the development of the product, for the final solution to be universally designed.

[1] Doran 1981: 36.

[2] Stene, 1999: 67.

### 3.3 Process Goals

It is possible to argue that the purpose of this project, at an overall level, is to provide training for the students in system development, while working agile in teams with an external client. Considering this the impact goal is therefore to increase the members competences in team collaboration and developing individual programming capabilities. In order to achieve this, the team must follow the guidelines reached in the collaboration agreement by supporting each other towards the joint goals.

## 4. Stakeholder and User Descriptions

For our project to be competitive within the budgeting and accounting market, we must understand the needs of our users and provide them with a tailored product. This is achieved through conducting research and collaborating with stakeholders to gather their requests.

### 4.1 Market Demographics

The application targets households and individuals in need of an easy-to-use application for budgeting and accounting a household economy. The specific market we target would be those who are in the market for a user-friendly, yet powerful, desktop application for managing their finances.

While there is an abundance of Mobile Apps and Web-based services for budgeting, who require the creation of accounts, paying subscriptions and are limited by “web technologies”, there is a glaring lack of standalone applications in this category. We target those who are not convinced by the online applications and services.

The market is drowning for web-based budget “software”, which makes it very hard to stand out. By having an actual application, which can be installed on a local computer, we aim to make our users feel actual in control of their own data, and not be bothered by online accounts and subscriptions. While this may be a niche group, it is one that is mostly overlooked by the app industry.

### 4.2 Stakeholder Summary

In this section, we provide a brief overview of our project stakeholders, including their backgrounds and responsibilities.

| Name | Description | Responsibilities |
| --- | --- | --- |

| | | |
|---|---|---|
| Student team | Product developers | Develop the application according to client specifications and assignment requirements.<br>Contribute to the team and the project to the best of their ability.<br>Ensure that all work is completed and delivered within assigned deadlines.<br>Ensure that the final product meets client specifications and assignment requirements. |
| Daniel Andre Evensen | Teaching Assistant | Guide students by suggesting solutions and providing feedback on project progression. |
| Svein Magnus Walaas | Client and end user | Communicate desired final product features and application design with developer team.<br>Participate in product testing throughout development and provide feedback on user experience (UX), user interface (UI), and usability, as well as evaluate how well the current product aligns with the original vision. |
| Course Lecturer | End user | Evaluate the final product and offer feedback on project during team meetings. |

### 4.3 User Summary

This section provides a brief description of the users involved in the project, their responsibilities, and their key problems that need to be addressed by the proposed solution, along with their names and stakeholder classification.

| **Name** | **Description** | **Responsibilities** | **Stakeholder** |
|---|---|---|---|

| | | | |
| --- | --- | --- | --- |
| Client | Individual who has a personal connection to one of the team members and is interested in improving their financial situation | Manage income and expenses<br>Get overview of income and expenses | Self-represented |

### 4.4 User Environment

The target environment is a desktop/laptop personal computer. It will run offline, as a standalone application. It will run on Apple MacOS, Linux (target distro: Ubuntu and derivatives) and Microsoft Windows. The user environment requires a recent version of Java Runtime Environment to be installed on the local machine. The application has been developed using Java Development Kit 17 (LTS) to maximize compatibility and stability.

### 4.5 Key Stakeholder or User Needs

After meeting with our client, we have received the requested features that are to be added. These requests are just suggestions and might not be added.

| **Need** | **Priority** | **Concerns** | **Current Solution** | **Proposed Solutions** |
| --- | --- | --- | --- | --- |
| Add new type of income or new expenses | High | Recurring payments | None | |
| Categorize income and expenses | High | | None | |
| Manage income and expenses | High | | None | |
| Overview of income and expenses | High | Create graphic charts | None | |
| Manage receipts and other documents | Medium | Store documents | None | Receipts and other documents will be stored in a database |
| Dark mode | Low | | None | |

| | | | | |
|---|---|---|---|---|
| Password protected database | High | Security | None | |

### 4.6 Alternatives and Competition

A plethora of competing products are available. The vast majority of these are online and web-applications. The fact that we are developing a standalone desktop application makes us stand out in the crowd and enables recruitment of a group of users that are overlooked by the mainstream application industry; namely those who do not want to register accounts, pay subscriptions and accept EULA and “Privacy” policies and give away their personal information.

A few examples of competing products would be:

| **Name** | **Platform** | |
|---|---|---|
| YNAB (You Need A Budget) | Web | Subscription. Requires online account. |
| Honeydue | Mobile only | Free or subscription. Requires online account. |
| Pocket Guard | Web | Subscription. Requires online account. In-App Advertisement. |
| Intuit mint | Web, desktop | Subscription. Requires online account. |
| Google Spreadsheet | Web, Mobile | |
| Microsoft O365 Excel | Web | Online-Only. Subscription. Requires online account. |
| Microsoft Office Excel | Desktop: Windows, Mac OS | Offline files. Integrated Cloud (can be disabled). Subscription required. |
| LibreOffice Calc | Desktop: Windows, Mac OS, Linux | Offline files. No cloud integration. |
| GNUCash | Desktop: Windows, Mac OS, Linux | Offline. No cloud. Free. F/LOSS. |
| KMyMoney | Desktop: Windows, Mac OS, Linux | Offline. No cloud. Free. F/LOSS. |

From the table above, we can identify our main competition as either the time-tested offline

spreadsheets (Excel, Calc), or the offline applications (GNU Cash, KMyMoney).

In order to make a convincing argument for our application being a better alternative, we must make it more attractive in terms of user experience than a spreadsheet and be at least on-par with the dedicated desktop application in terms of usability and features. Even if we are not able to match them in the depth of advanced features, we should be able to make our core features so polished and easy to use that it will be worth the loss of some more advanced options.

# 5. Product Overview

The product will be a tool for managing personal finance, deployed as an offline desktop application. It will enable the user to manage budget and accounting for a household.

## 5.1 Product Perspective

The product is a standalone desktop Java application, and as such will be able to run on most modern desktop computers. Outside the need to interface with the local machine's file system, there will be no need for the application to be considered a part of any other system. The instances in which the interfacing will be required would be limited to:

- Storing application data on the local file system.
- Locate files in the computer file system and load these into the application's archive.

This interfacing will be handled via the JVM and .jar interaction with the local computer.

## 5.2 Summary of Capabilities

The table below gives an overview of how the application aims to benefit the client. It makes some assumptions regarding features being planned.

| **Customer Benefit** | **Supporting Feature** |
| --- | --- |
| Easy overview of budgetary constraints. | - Overview of expenses and income.<br>- Visual information via charts and trend lines. |
| Visualize financial status via easy-to-read charts and trend lines. | - Various charts and trend lines. |

| | |
| --- | --- |
| Management receipts and other documents, attached to a posting or as a single file in the Archive section. | - Archive function.<br>- Manage documents via the application. |
| Privacy and security. | - Offline application.<br>- Local installation.<br>- Data stored locally.<br>- Password protected database.<br>- No data is ever transmitted outside the local machine. |
| Portability. No vendor Lock-In. | - Data can be exported in an Open and portable format, as stipulated by GDPR.<br>- Can be imported into any application supporting the exported format. |

*Table 4-1: Application Capabilities*

### 5.3 Assumptions and Dependencies

**Operating System:**

- The application will be written in Java, and as such be platform-agnostic.
- I will run on Mac OS, Linux and Windows, all of which are represented among the development team.

**Software Requirements and Dependencies:**

- The application will require the installation of Java Runtime Environment (JRE).
- This will need to be downloaded and installed, either via the Java home page (Windows, Mac OS, Linux) or via a package repository (Linux).

**Computer System Requirements:**

- Any reasonably recent computer (desktop or laptop) should be able to run this program.
- The development team will be able to test the application only on the range of laptops and desktops they have available, which will limit the ability to test lower limits of requirements but are confident that our combined computers represent a

good average of what the average user would have.

### 5.4 Risk Analysis

The matrix below gives an overview of potential risks, as identified by the team. They have been assigned a Probability-Consequence score, based on their probability to occur, and the severity of the consequence should they occur.

| | | **Consequences** | | | | |
|---|---|---|---|---|---|---|
| | | Insignificant | Minor | Moderate | Major | Catastrophic |
| **Probability** | Highly probable | | | | | |
| | Likely | | F | B | | |
| | Possible | | G | | C | |
| | Unlikely | | E | | D | |
| | Highly improbable | | | | A | |

| **Risk acceptance criteria** | |
|---|---|
| **Level** | |
| **High risk** | The red colour represents risks that is not regarded acceptable, and measures must accordingly be taken immediately. |
| **Moderate risk** | The yellow colour alerts the needs for introduction of measures. These risks are not considered as urgent as the red ones, but the must nevertheless be taken seriously and attempted solved to the best of their ability. |
| **Low risk** | The green represents an acceptable risk. Action will only be assessed if the measure will have a positive effect on the project. |

**Risks identified:**

A. Limited development time. May result in a final product not meeting the task description of the course.

B. Incomplete or poorly executed features and functionalities due to lack of time

C. Limited testing resources due to limited development time

D. Limited knowledge in programming within the team

E. Lack of feedback from client, due to availability.

F. A team member becomes unavailable due to illness or similar circumstances.

G. Disagreements and conflict between team members impede progress.

Each of the potential risks has been discussed and evaluated internally within the team. This making them recognizable to all, so that necessary measures can be taken when required. The evaluation is based upon the fact that there is some probability attached to all of them. Nevertheless, as illustrated in the matrix, most of the potential risks were considered as moderately risky. Although they were not considered as urgent, they could still be significant for the progression of the project. Hence the internal acceptance of the team's milestones and communication will serve as two important factors in handling these kinds of risks. Nonetheless, this also applies to the potential risks that are beyond the team's control. Issue E, F and G can be highlighted in this context. In these cases, planning and communication can impede them from occurring, and thereby minimizing its negative impact on the final product. Due to the diversity of potential risks, it would be appropriate to account for them individually in the following sections:

The team has identified limited development time as one significant potential risk affecting the final product while working on the project. This limitation can be realized as a result of various reasons, for instance unforeseen challenges, low work efficiency and the realization of the other risks, to name a few.

**Risk A** have consequences directly linked to the final product not meeting the given task description. As this will result in a final product not working as intended, and therefore arise as a major consequence. For instance, if the product is lacking guidelines for economic improvements or not meeting the WCAG standards, as set in the project's requirements. This can further be justified by the fact that a lacking final product will not be graded highly, and therefore not meeting the team's result goal. However, the risk is considered highly improbable, as a result of the agreement to constantly communicate and make plans to focus on product functionality rather than desired features. This way prioritization of necessities can ensure that the requirements in the task description are met.

**Risk B** is about how a lack of time can lead to incomplete or poorly executed features and

functionalities, resulting in a final product that might not meet the client's expectations or requirements. This risk is regarded as likely to happen, because the student team doesn't have any previous experience with larger projects, and therefore may have agreed upon a project scope that is too big to succeed. The consequences of the risk occurring are considered moderate, because the final product would still be viable, but lack level of quality, features and polish, leading to customer dissatisfaction. To prevent risk B from happening, the team should break the project into smaller, more manageable tasks, and set clear goals and milestones for each scrum sprint. They can also involve the client in the prioritization process to ensure that the most critical features are delivered on time. If a deadline has passed without the intended goal being reached, the team must reconsider the scope of the project and possibly make changes.

**Risk C**, which is limited testing resources due to limited development time, can result in undiscovered bugs in the final product. This would have major consequences for the project because bugs and error could lead to user frustration and the application crashing or not working as intended. Risk C is considered probable, because testing often gets overlooked or reduced in projects with limited time and resources. To reduce the chance of risk C occurring, the team should use available resources efficiently and prioritize testing efforts strategically. For instance, the team should write test classes using JUnit, because it can be used to automate the testing process and reduce the time and resources required for manual testing. If risk C becomes a reality, the team must prioritize their testing efforts based on the most critical features and functionalities. They should also conduct a risk assessment to identify the most severe and probable issues and address them as soon as possible.

**Risk D** is a significant risk that can impact the success of the project, which is limited knowledge in programming within the team. If team members lack the necessary knowledge and experience in programming, it can lead to poor programming decisions and ineffective solutions. Early bad choices in programming can also make it difficult to expand the codebase in the future. The consequences of the risk happening can be major, resulting in delays and reworks. Moreover, the delays can have a cascade-effect, where for example feature X requires feature Y to be complete before it can progress. Although the consequences of risk D are major, it is unlikely to happen, since the team can seek guidance from each other or the teacher assistant to help make effective programming decisions. In addition, a class diagram will have been made and discussed before the programming begins. If risk D happens, the team must seek advice from the teacher assistant. Another option is to review and refactor existing code to improve its quality. The team can also conduct throughout testing to identify and fix any issues or bugs that might arise from programming mistakes.

**Risk E** the lack of client feedback is another risk identified by the team. As opposed to the rest of the risks, this is the only one regarded as a low risk due to its minor consequences and minor probability of occurrence. This evaluation can be justified by our experiences with the client thus far, and the relation we established through the meetings we have performed. If the contact with the client were to fade due to availability or some other

occurrence, the consequences for the final product would mostly impact user testing and external feedback for the design and feature set of the application (both MVP and final product). Pro-active communication and planning can be emphasized as a solution to this potential risk.

**Risk F** is related to a team member becoming unavailable due to illness or similar circumstances. This risk is likely to happen and can have some minor consequences, such as delays in project delivery, decreased productivity, and increased workload on the remaining team members. To minimize the risk, it's essential for the team to maintain clear documentation on ongoing tasks and responsibilities and communicate effectively. This way, in the event of a team member becoming unavailable, another team member can easily step in and take over their responsibilities without causing any delay or issues. If a member reports that they will be absent for a longer period, the team should immediately assess the impact of the absence and adjust the project timeline accordingly.

**Risk** G deals with disagreements and conflict between team members that impede progress are considered as possible, due to the composition of five individuals coming together as a joint team. In the working on the project, disagreements will naturally arise within the team. This can be understood as a healthy part leading up to the final product because the collaboration agreement allows for discussion if it results in benefits for the project, not just as a distraction. This risk is set to a low consequence, as the collaboration agreement also states clear guidelines for conflicts impending the work progress. For instance, chapter 5.A *General*, addresses the measures evoked by disagreements and internal conflicts by the use of majority rule and veto. These actions can in combination with sincere communication be used as guidelines in case of potential disagreements and conflicts within the team.

## 5.5 Quantifiable and Non-Quantifiable Benefits

Since this is a student project, there will be no hard, quantifiable benefits outside that of having a working application for budgeting and accounting.

One can argue that there is a quantifiable quality to the knowledge that each team member will acquire through the work put in, in comparison to what one knew prior to the project. We argue that this is the primary benefit of the project.

## 5.6 Licensing and Installation

The Application will be released as Free/Libre Open-Source Software (F/LOSS), under the GNU Public License v3.0 (GPLv3).

# 6. Product Features

The development team have, through meeting with the client and team-internal discussions, arrived at a set of basic functionalities we agree should be present in the final application. The features listed below are not final, and subject to change according to the progress of the project and by agreement with the client.

## 6.1 Standalone Desktop Application

- Offline application, running locally on the user's laptop or desktop computer.
- Password protected database.
- Ability to export the database, or load/import data from external database

## 6.2 Budget Overview

- A view which shows both monthly and annual budgeted income and expenses.
- Budget overview, both monthly and annual view.
- Data should be represented both with cold, hard numbers, and by easy-to-digest visual methods such as pie charts, bar charts and trend line graphs.
- Recurring payments.

## 6.3 Accounting and Expenses

- A view, separate from the main budget, where expenses can be added.
- Option to append files (such as a PDF) to an entry.
- Adding expenses on a rolling basis. Spending will appear in a summary.

## 6.4 File Archive

- Overview of all files (PDF, etc.) appended to entries, with relevant information to identify the entry and content.
- Section for miscellaneous documents not tied to a specific entry.

### 6.5 Recurring Payments

- Ability to add a payment and define it as a recurring. Weekly, monthly and yearly.

### 6.6 Dark Mode

- Dark colors in the interface, to reduce eye strain.

### 6.7 Visual representation: Charts and trends

- Financial situation should be shown “at a glance” via charts.
- Trend lines, tracking spending.

## 7. Constraints

The team have identified several constraints, and it’s important to keep these in mind as we plan the progress of our project. The constraints present themselves in several forms, such as the project specifications, technical limitations, time limitations, and the available programming knowledge within the team.

**Technical constraints** of the application will mostly be that it is an offline desktop application. As such, there will be no login to external services, nor any need to adapt the interface to a mobile device. This also frees us from any concern regarding privacy regulation, as there will be no personal data leaving the User’s desktop system.

Other technical constraints will be that it is to be made in Java, with JavaFX as the GUI toolkit. As such, dependencies for running the application are mostly the Java Runtime Environment.

**Time constraints** lie mostly in the deadlines provided in the assignment. They stipulate the time allotted for certain project milestones to be met. As such, we have a limited time in which to complete the assignment and must plan and prioritize accordingly.

**Human constraints** present themselves most prominently in the lack of Programming experience across the team. Therefore, the team will need to expand their programming abilities during the project. This will consume time that could overwise be used for working on the application. The constraint will be in balancing education with actual progress in building the application, and it is imperative that the members are honest and open about any difficulties or needing assistance.

There will also be constraints in how much access we have to the client for feedback and

testing, in addition to a lack of resources for performing a wide set of user tests, hardware tests and quality assurance.

**Design constraints** are those imposed by the Client, as well as those specified in the project assignment. We must strive to achieve WCAG compatibility, as well as meet our client's wishes with regards to UI, UX, Usability and Accessibility.

## 8. Quality Ranges

Usability is a very important aspect of modern application development, and the initial impression of quality is essential for the likelihood of a potential user sticking with an application. As such, we will have a steady focus on providing easy-to-read interface elements. To achieve a good sense of quality, we have identified certain key quality aspects:

- Be able to handle a large database of budgetary information without suffering performance loss. The application should feel smooth and responsive to use.
- Be protected from data loss and must be tested for this. We will not accept the loss or corruption of data. The error handling must be robust and thoroughly tested.
- Have a user-friendly, reasonably attractive and intuitive layout.
- Be rooted in universal design, and solid design principles.
- Be tested as thoroughly as possible to ensure (ideally) no bugs or errors in the finished application.

## 9. Precedence and Priority

The team has arrived at the following features as priorities:

| **Feature** | **Priority** | **Importance** |
| --- | --- | --- |
| Budget overview | 1 | Highest |
| Accounting overview | 1 | Highest |
| Expense and income | 1 | Highest |
| Visualized data (charts, etc.) | 3 | Medium, Nice to have |
| File archive | 2 | Important |
| Dark Mode | 4 | Low, Nice to have |

# 10. Other Product Requirements

## 10.1 Applicable Standards

- The application will adhere to ISO standards.
- The first day of the week is Monday, as according to ISO-8601
- Dates are especially important in a budget and accounting program. Thus, it will be displayed in one of two standardized ways.
    - ISO-8601, Technical archival format: “yyyy-mm-dd", e.g.:
        - 2023-01-18
    - ISO and European “normalized” standard: “dd-mm-yyyy", e.g.:
        - 18-01-2023
        - Or formatted as: 18 January 2023
- Time will be displayed in the international standard 24-hour format.
- Thousand separators shall be a hard space, as per ISO standards and B*ureau International des Poids et Mesures*, Resolution 10 of the 22nd CGPM (2003):
    - ***declares*** *that “the symbol for the decimal marker shall be either the point on the line or the comma on the line”*
    - ***reaffirms*** *that "Numbers may be divided in groups of three in order to facilitate reading;* ***neither dots nor commas are ever inserted in the spaces between groups****", as stated in Resolution 7 of the 9th CGPM, 1948. (Emphasis ours)*
- Decimal separator shall be “comma on the line”, as per ISO 80000-1.
    - *The standard does not stipulate any preference, observing that usage will depend on customary usage in the language concerned, but adds a note that* ***as per ISO/IEC directives, all ISO standards should use the comma as the decimal marker****. (Emphasis ours)*

## 10.2 System Requirements

- The application will run on the operating systems Apple Mac OS, Linux (target distro: Ubuntu, and derivatives), and Microsoft Windows.
- It will require Java Runtime Environment JRE 17 (LTS).

# 11. Documentation Requirements

The documentation of use must stand in relation to the scope of the project. As such, a User Manual and guide for configuration and installation will be required. Anything beyond this would exceed reasonable levels of documentation for an application of our intended scope.

## 11.1 Internal Documentation

To facilitate easier collaboration, and to make it easier to plan ahead, not to mention finalizing the project report, all source code, models, mock-ups, meeting reports and other documents produced during the project is to be stored on the team's cloud drive. During the later stages of the project, this documentation is to be presented in the project Wiki page on GitLab.

Source code will adhere to best practices for code documentation (comments and JavaDoc). It should be written in a precise and clear manner.

Documentation shall be performed in parallel with the project work

## 11.2 User Manual

The User Manual will give the user an easy Get Started guide to the basic functionality of the application, in addition to information about the more advanced features. The manual must describe how these features, both basic and advanced, are intended to be used and how they will improve the usefulness of the program.

## 11.3 Online Help

A dedicated Wiki will be made available on the projects GitLab page, tied to the project repository. All documentation relevant to the function of the application will be made available there.

## 11.4 Installation Guides, Configuration, and Read Me File

Readme-files for configuration and installation will provide clear and simple instructions for how to set up the initial installation of the application, e.g. dependencies needed and how to build the software if compiled from source, in addition to providing a short troubleshooting section, usually of known issues a user may have during initial installation and setup, issues with compatibility.

### 11.5 Labeling and Packaging

The project does not use any copyrighted or patented labelling. GUI icons and help systems will use the icons and labeling provided by the service or toolkit. Other logos, artwork, etc., will use either Original Content, or items released under a permissive license or in the Public Domain.