uc3m | Universidad Carlos III de Madrid

Software Development Project Management

Date: 25/02/2022 Group: 2

Alvaro Morata Hontanaya
Brian Jesse Gatukui Kimani
Ainara Machargo
Maria Opland
Justin Pimentel
Elvis Vasquez
Tommy Woldseth

Client: (name of the company)

Content

Tables index

No se encuentran elementos de tabla de ilustraciones.

Figures index

No se encuentran elementos de tabla de ilustraciones.

CREATIVE IDEA DESCRIPTION

Burn-out in the workplace is a growing issue in occupations all over the world. It is defined as a syndrome resulting from workplace stress that has not been managed, according to WHO (*Burn-Out an "Occupational Phenomenon": International Classification of Diseases*, 2019). One of the reasons people experience burn-out is that workers do not feel rewarded for their efforts (Aumayr-Pintar et al., n.d. p.17). In addition to that, another reason for burn-outs is working too much without a sufficient amount of breaks. To combat this problem we have come up with an idea in the form of an app intended for employees, called BonusBreak. The main feature of BonusBreak is to reward employees for taking breaks and encourage teams to have a healthy work environment.

BonusBreak seeks to create a unique experience that enables each individual user to track their mental wellness in the workplace, while still creating incentive on the team level to encourage a healthy work environment. Our main target market is teams mostly in software engineering, that rely on repeating cycles, for example two-week development sprints, to accomplish work tasks for a project in broken down stages. On an individual level, users of the app will answer a brief survey during sign up that describes their current workstyle, means of de-stressing, interests outside of work and any current causes of stress, anxiety and unrest in the workplace. With this data, we plan to tailor specific routines for each user that the app will recommend they apply to their day-to-day work habits. The routines will include scheduled break reminders with suggested activities to partake in during these breaks. The activities will focus on a user's preferred interests and ways of de-stressing. Through our platform, a user will be able to visualize and keep track of how implementing various recommended tasks affects their work style, productivity and improves overall wellness. Throughout workdays, users get pinged with spontaneous notifications from the app to describe their current feelings which will contribute to this data.

At the individual level, employees will receive points based on how they work, e.g. if they have an effective working session in between breaks. Furthermore, the employees will earn points by taking breaks, up to a certain limit. Since taking breaks is an important part of reducing burn-out, the combination of effective working sessions and taking breaks will be essential. In addition to regular breaks, employees will be introduced to three daily challenges at the beginning of the work day that will result in points upon completion. These challenges will

e.g. be socializing, give a coworker a compliment, say hello to five people at the workplace etc. Participating in the app will be optional to avoid putting pressure on the employees. To further motivate the workers to be consistent, users will receive bonus points for obtaining a streak of using the app in consecutive days.

At the team level, BonusBreak incentivizes these recommended routines by utilizing a reward system that we define as a work-life balance ratio meter. The metric would keep track of the time a user spends throughout the day on work related duties vis-à-vis their break times and recommended activity completion. The ratio uses a point system that collectively rewards teams for spreading out of work throughout the day with the recommended number of breaks in between work sessions being achieved. Points are also rewarded for completing daily tasks, which allows the users to focus on something other than work and encourages them to socialize with other team members. Before the start of an iteration, the team manager sets a budget for the reward within the app, as well as the number of points needed for the team to get the reward. The employees can then submit suggestions for what type of rewards they would like. As being social is proven to reduce burn-out, the rewards should be different types of social events, like for example office parties, cabin trips, dinner events etc. (Aumayr-Pintar et al., n.d. p.16). Optionally, the reward could also be a monetary bonus or time off, if the team would rather want that. To decide on the reward for an iteration, the employees would vote through the app on the reward that they want. The votes of the employees would be valued according to the number of points that they received during the iteration, to further motivate taking enough breaks. As the employees are not able to see how many points others have accumulated, there would not be unnecessary pressure to earn a lot of points, but with good rewards the users would feel motivated to help their team reach the goal. The employees would also be able to see their team's point total, and the app would display statistics about whether the team is behind or ahead of schedule to reach the iteration goal.

1. General data of the company offering the project

- **Name**: BonusBreak
- **Acronym**: BB
- **Description**: New software company that wants to help employees all over the world to live a more comfortable work-life balance, and have higher satisfaction from their jobs.
- **Mission**: Reduce burn-out in the workplace

2. Definitions and acronyms

- Work-life balance ratio meter: Measure of the balance between work and breaks for an user of BonusBreak
- Points: A measurement system that will reflect a user and a team's performance in the work aspect, taking breaks highly into consideration, and the user's social performance, by participating in the daily tasks. In addition, it will reflect the frequency that the user is on the app, and more points will be rewarded for multiple-day streaks.
- Rewards: Points generated by a team can be redeemed for a reward, such as work retreats, monetary bonus, or days off. Rewards can be suggested to the team manager through the app.

3.Initial offer and budget

3.1 Offer

3.1.1 Background

This project is targeted to companies within the technological sector. Any company with a medium to large number of employees can benefit from BonusBreak's advantages, effectively reducing the burnout in the workplace and enhancing the productivity and happiness of the employees, not only to a personal level but also to a team level.

3.1.2 Analysis of the context

Work-fit is one of our primary competitors as they offer a wellness management program intended to provide workers access to professionals that will help to improve their wellness with activities such as mindfulness therapies, yoga sessions, etc. BonusBreak differs as it focuses on both the individual and team level. On an individual level, employees receive incentives by gaining points for doing things that will reduce the likelihood of burnout e.g taking breaks, complimenting coworkers for positive stimulation, and receiving bonus points for consistently using the app. On a team level, we are utilizing a reward system that incentivizes a work-life balance ratio by tracking the time an employee spends working throughout the day, amount of tasks completed, and giving employees the option of voting on what rewards they would like—whether it be social like a dinner or trips or just time-off/monetary.

3.1.3 Main objective

The main objective is building a system that is able to prevent burn-out syndrome through some of the known methods that allow workers to relieve stress and improve their wellness in the workplace.

3.1.4 Scope of the project

Bonus Break aims to prevent burnout syndrome from becoming a big issue within the workforce. If it already exists within the office, then its aim will be to reduce it. It can reduce and prevent it simultaneously. The main limitation of our project would be that it can only work if both the employees and team managers are making the effort to combat burnout by using the application. With the model in place, Bonus Break should provide the incentive needed for employees to reap benefits as they actively combat burnout and to promote a more collaborative work environment which will also fight burnout in the long run and promote the use of the application as employees would want to collaborate.

3.1.5 Action plan

Below is included a description of the different phases of the project and a graph showing how they are distributed:

Phase 1: Management and monitoring
- Gather the initial resources.
- Manage and monitor the project during the different stages.

Phase 2: Analysis

Phase 3: Development
- Design phase.
- Implementation phase.

Phase 4: Data loading and validation
- Verification and validation.
- Deployment.

To get the estimation of the amount of time the project will take we have used the number of use cases in the use case diagram. By dividing the number of use cases by 4 we get an estimation of the number of months that will take to complete the project. Considering we have 23 use cases, and two of them has at least 4 transactions, we get the following:
- 21/4 = 5,25 months for simple use cases
- 2/4 * 1,5 = 0,75 months for average use cases
- Total time to complete the project: 6 months.

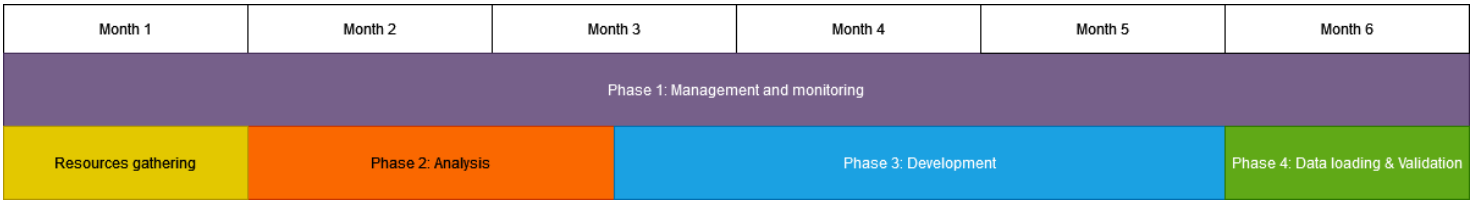


Figure 1: action plan

3.1.6 Teamwork

Below are included two graphs detailing the number of software engineers and programmers employed in each phase. Notice that phase 3 has been separated into the design and implementation phases.

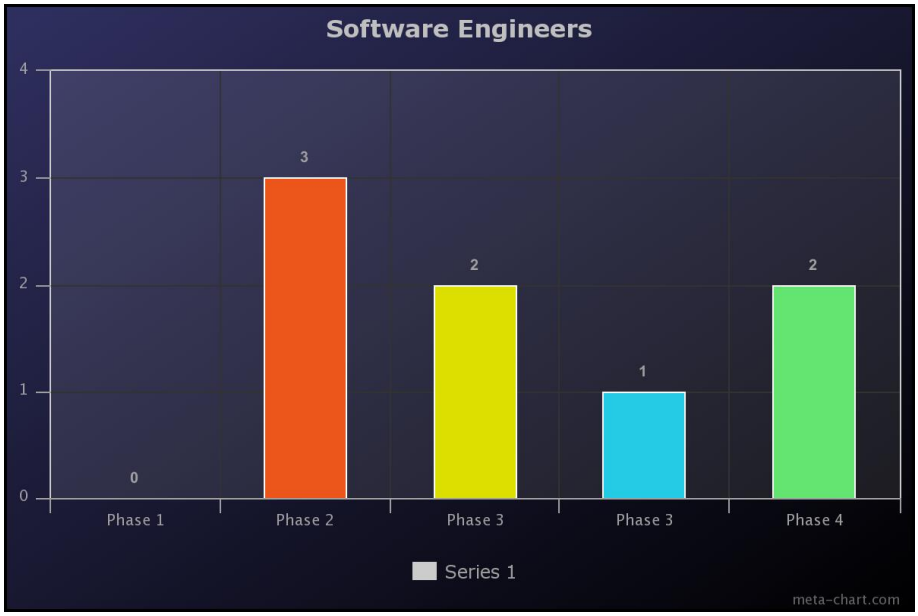


Figure 2: graph of the number of SE per phase

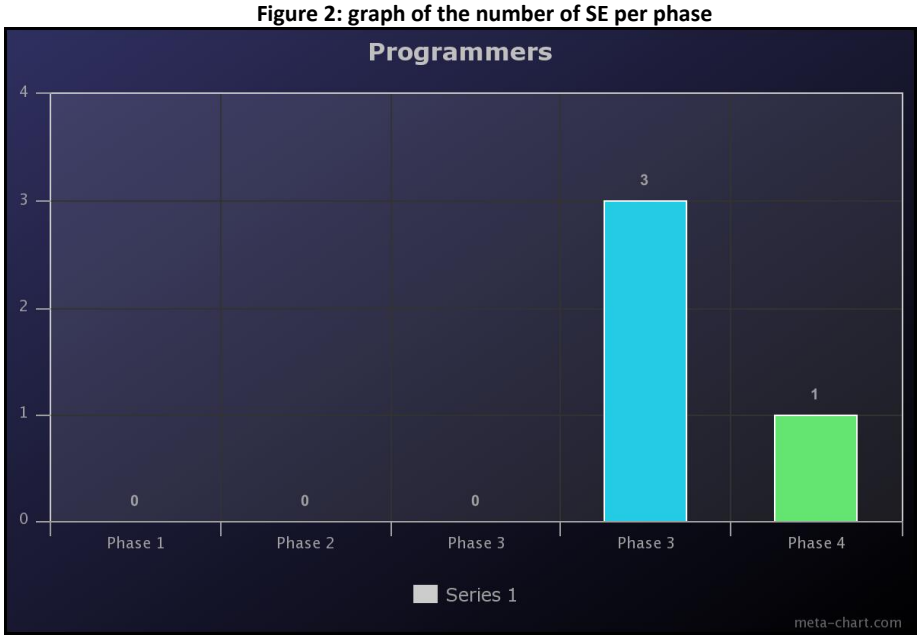


Figure 3: graph of the number of programmers per phase

3.2 Budget

3.2.1 Salaries

Taking into consideration the 23 use cases, where 21 of them are simple and 2 of them are average use cases, we got from the action plan a total amount of 6 months of project development.

The gross salaries have been based on the averages in Spain. These salaries are as follows:

- Software engineer: 2.800€ per month.
- Programmer: 2.300€ per month.

Both job positions' salaries are based on a 8 hours per day, 5 days a week model.

Breakdown of the salaries:

- Software Engineers.
  - Phase 2:
    3 engineers * 1’5 months * 2.800€/month = 12.600€
  - Phase 3:
    2 engineers (design) * 1 month * 2.800€/month + 1 engineer (implementation) * 1’5 month * 2.800€/month = 9.800€
  - Phase 4:
    2 engineers * 1 month * 2.800€/month = 5.600€
  - Software Engineers subtotal: **28.000€**
- Programmers.
  - Phase 3:
    3 programmers * 1’5 month * 2.300€/month = 10.350€
  - Phase 4:
    1 programmer * 1 month * 2300€/month = 2.300€
  - Programmers subtotal: **12.650€**
- **Total: 40.650€**

3.2.2 Table of the budget

Description	**Total**
Salary	40.650€
Computer equipment	3.300€
Software	800€
Consumables	360€
Travel expenses	3.900€
VAT (21%)	10.292,10€
Profit (10%)	5.930,21€
Risk (15%)	8.895,32€
TOTAL	**74.127,63€**

Figure 4: Table of the budget

First payment: 29.651,05€
Second Payment: 44.476,58€

4.Software Configuration Management Plan

INTRODUCTION

4.1 Purpose of the Plan

The Plan detailed below is aimed at both the development staff and the management team. The aim is to make the project sufficiently robust to collect

information about the state of the product and to make a change. The changes are especially delicate in this one, since there are elements that require special attention and care when modifying them.

It is therefore intended to document each baseline and each change made as indicated below when detailing configuration management activities.

4.2 Scope

This SCM plan will apply to the project BonusBreak.

4.3 Definitions and Acronyms

The following are the acronyms used in this Configuration Management Plan:

SCM - Software Configuration Management

UML - Unified Modeling Language

CE - Configuration Element

4.4 References

Aumayr-Pintar, C., Cerf, C., & Parent-Thirion, A. (n.d.). *Burnout in the workplace: A review of data and policy responses in the EU*. EU Agenda. Retrieved February 18, 2022, from https://euagenda.eu/upload/publications/untitled-178342-ea.pdf

Burn-out an "occupational phenomenon": International Classification of Diseases. (2019, May 28). WHO | World Health Organization. Retrieved February 18, 2022, from https://www.who.int/news/item/28-05-2019-burn-out-an-occupational-phenomenon-international-classification-of-diseases

Cross-platform mobile frameworks used by global developers 2021. (2021, July 16). Statista. Retrieved March 6, 2022, from https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/

MANAGEMENT SPECIFICATIONS

This section identifies the coordination and management tasks that will be necessary to carry out the SCM.

4.5 Organization

There must be permanent and direct contact between the development staff and the change control committee, so that delays in the processing of a change are

as short as possible, so that both improvement and correction processes are not tedious work.

Both the change control committee and the other development staff should pay special attention to the points where it has been stipulated that baselines will be established within the development. For more information see the section on Definition and Establishment of Baselines**.**

4.6 Responsibilities

Change control committee: Brian Jesse Gatukui Kimani, Elvis Vasquez, Álvaro Morata Hontanaya

Responsible for SCM: Ainara Machargo

Librarian: Maria Opland

Rest of the development staff: Tommy Woldseth, Justin Pimentel

4.7 Applicable policies, directives and procedures

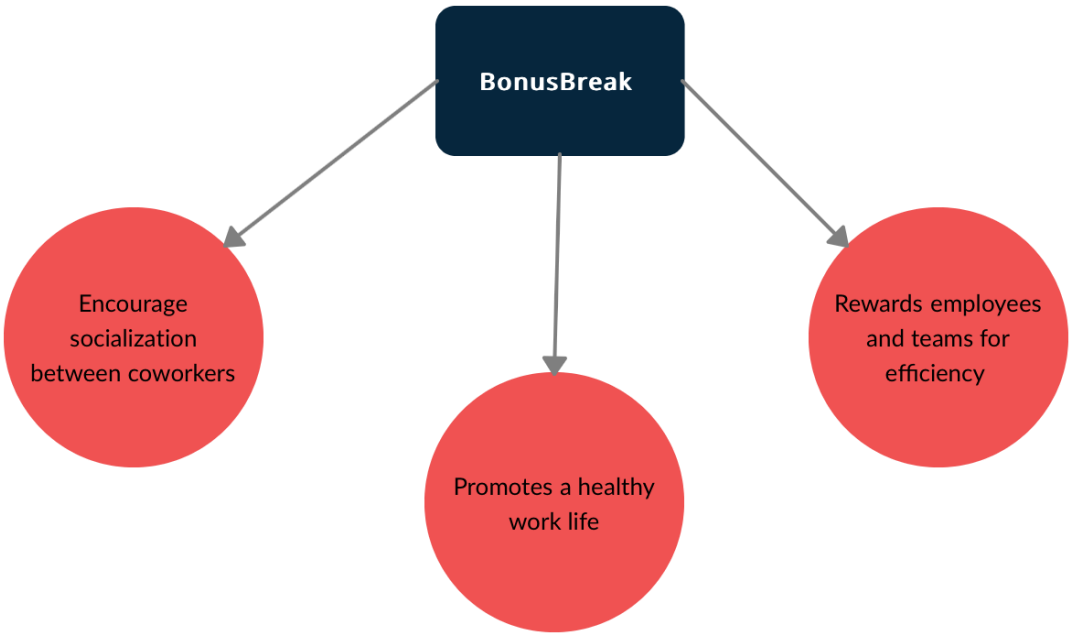
The applicable procedures are described in the section: "Configuration Change Control".

CONFIGURATION MANAGEMENT ACTIVITIES

The following is a description of the SCM activities that will be carried out during the development of this project.

4.8 Configuration Identification

4.8.1 The preliminary product hierarchy is established



first overview of the structure and elements that the software system will have.
Slide 36

Make small graph

Figure 5 General system structure

This app will encourage socialization between coworkers, a healthy work life, & proper rewards for employees.

4.8.2 Selection of the configuration elements

All configuration elements:

- Offer
- Budget
- Quality Plan
- SCM Plan
- SCM Plan Review
- First draft of the use case model
- Estimation
- Estimation review
- Schedule
- Schedule review
- Feasibility analysis (including requirements specification)
- Feasibility review
- Use cases model
- Use cases model review
- Prioritization of use cases
- Prioritization of use cases review.
- Definition of high-level use cases.
- Definition of high-level use cases review
- Use cases in extended format.
- Use cases in extended format review
- Conceptual model
- Conceptual model review
- Operation Contracts
- Operation Contracts review
- Class diagram
- Class diagram review
- Sequence diagrams
- Sequence diagrams review
- Transition states diagram
- Transition states diagrams review

4.8.3 Selection of the identification scheme

CE code	Name	Desc.	Date	project	Baseline	Type	Appointee

Figure 6: Template for the identification scheme

4.8.4 Definition of relationships

Dependency

CE 1	CE 2	Date

Derivation

CE 1	CE 2	Date

Succession

CE 1	Previous Version	Next Version	Date

4.8.5 Definition and establishment of baselines

Our baselines are as follows:

- Phase 0: Initial planning.
- Phase 1: Planning and requirements specification.
- Phase 2: Construction Phase.
    - Iteration 1
        - Analysis
        - Design
        - Coding
        - Testing
- Phase 3: Installation phase.

The purpose of these baselines is to offer the team a possibility to make any informal changes before the baseline, but also any formal changes once the processes are done, allowing the team to constantly update and maintain the project and ensure that the phases are going as planned.

4.8.6 Definition and establishment of software libraries

As BonusBreak is a native mobile application we would either have to make one Android version and one iOS version, or we could use a hybrid framework to make one version that works for both operating systems. We have chosen to use React Native which is one of several available hybrid frameworks for mobile application development. React Native is an open source JavaScript framework that allows us to develop and maintain one codebase instead of two. This will keep costs for the project down, and productivity of the developers up. It is also possible to make a prototype quickly with React Native, which is very beneficial when using an iterative methodology. It is one of the most popular hybrid frameworks, which leads to a big development community and a lot of helpful additional packages (*Cross-Platform Mobile Frameworks Used by Global Developers 2021*, 2021). The documentation for React Native is available here: https://reactnative.dev/docs/getting-started. React Native is MIT licensed, which allows us to sell software that is based on it.

4.9 Changes control

It is requested:

APPLICABLE CHANGE CONTROL PROCEDURE

1. Initiation of change: the request for change, duly completed by the applicant, shall be submitted.

2. Classification and registration of the request for change.

3. Evaluation and Approval or rejection by the Change Control Committee.

4. In case of approval, notification to the originator and to the managers of the CEs concerned.

5. The change is made by entering a monitoring and control process.

6. Once the change has been made, the change control committee certifies that it has been made correctly.

7. Finally, the originator of the change is notified of this certification.

- Change Request Report Format

<table>
<tr><th colspan="3">Change Request Report</th></tr>
<tr><td colspan="2">System name:
System code:</td><td>Level of implementation of change:
System:
Hardware:
Software:
Documentation:
Another one:</td></tr>
<tr><td>Name of the applicant:

Phone:

Date of application:</td><td>Priority of change:
Routine:
Urgent:
Very urgent:</td><td>Are other hardware or software systems affected?
YES
NOT</td></tr>
<tr><td colspan="3">Description of the change:

Need for change:

Estimation of the effect of the change on other systems, software and equipment

Alternatives to change:</td></tr>
<tr><th colspan="3">To be filled in by the change control team</th></tr>
<tr><td>Date of receipt of application:</td><td colspan="2">Disposition:</td></tr>
<tr><td colspan="2">Signed:</td><td>Date:</td></tr>
</table>

Figure 7: Report 1 Change Request Report

- Change Certification Report Format

Certification of the change:
Date of certification: **Originator:** **Recipient:**
Results obtained Signed:

Figure 8: Report 2 Change certification report

4.10 Status account

Configuration Items:
Acronym meanings:
Development Processes: DP
Management Processes: MP
Control Processes: CP
Offer: O
Budget: B
Use Case Model: UCM
Software Configuration Management Plan: SCMP
Quality Plan: QP
SCM Plan review: SCMPR
Feasibility Analysis: FA
Feasibility Analysis: FAR
Use cases model: UCM
Use cases model review: UCMR
Prioritization of use cases: PUC
Prioritization of use cases review: PUCR
Definition of high-level use cases: HLUC
Definition of high-level use cases review: HLUCR
Estimation: E
Estimation Review: ER
Schedule: S
Schedule Review: SR
Use cases in extended format: UCEF
Use cases in extended format review: UCEFR
Conceptual model: CM
Conceptual model review: CMR
Operation Contracts: OP
Operation Contracts review: OPR
Class diagram: CD

Class diagram review: CDR
Sequence diagrams: SD
Sequence diagrams review: SDR
Transition states diagram: TSD
Transition states diagrams review: TSD

CE code	Name	Desc.	Date	Project	Baseline	Type	Appointee
MP-O	Offer	Describes the project and project goals	2/25/22	Bonus Break	Phase 0	Document	Justin
MP-B	Budget	Describes the amount of money that will be needed to develop the project	2/25/22	Bonus Break	Phase 0	Document	Alvaro
DP-UCM	use case model draft	illustrates the interactions within the project	2/25/22	Bonus Break	Phase 0	Document	Ainara
CP-SCMP	SCM plan	Identifies everything that will need to be controlled throughout the process	3/7/22	Bonus Break	Phase 0	Document	Elvis & Brian
MP-QP	Quality plan	Identifies what to review and how	3/7/22	Bonus Break	Phase 0	Document	Tommy & Maria
MP-SCMPR	SCM Plan Review	Verifies the SCM plan is correct	3/7/22	Bonus Break	Phase 0	Document	Elvis & Alvaro
DP-FA	Feasibility analysis	Defines the requirement	3/14/22	Bonus Break	Phase 1	Document	Ainara & Justin

	(including requirements specification)	s to deduce the feasibility of the project					
MP-FAR	Feasibility analysis Review	Verifies the Feasibility analysis is correct	3/14/22	Bonus Break	Phase 1	Document	Tommy
DP-UCM	Use cases model	defines the use cases that will be used	3/28/22	Bonus Break	Phase 1	Document	Tommy & Brian
MP-UCMR	Use cases model review	Verifies the Use cases model is correct	3/28/22	Bonus Break	Phase 1	Document	Elvis
DP-PUC	Prioritization of use cases	Prioritizes which use cases are to be processed first	3/28/22	Bonus Break	Phase 1	Document	Alvaro
MP-PUCR	Prioritization of use cases review.	Verifies that the prioritization of use cases reviews are correct	3/28/22	Bonus Break	Phase 1	Document	Justin
DP-HLUC	Definition of high-level use cases.	Describes the use cases more in depth	3/28/22	Bonus Break	Phase 1	Document	Maria
MP-HLUCR	Definition of high-level use cases review	Verifies the definition of high-level use cases are correct	3/28/22	Bonus Break	Phase 1	Document	Ainara
MP-E	Estimation	Quantifying the efforts used based on the Use	4/19/22	Bonus Break	Phase 0	Document, Excel sheet	Brian

		cases					
MP-ER	Estimation Review	Verifies that the estimation is correct	4/19/22	Bonus Break	Phase 0	Document	Elvis
MP-S	Schedule	Details the planning process	4/19/22	Bonus Break	Phase 0	Document, Microsoft project	Alvaro
MP-SR	Schedule Review	Verifies that the schedule is correct	4/19/22	Bonus Break	Phase 0	Document	Justin
DP-UCEF	Use cases in extended format	Provides more details about the use cases	5/6/22	Bonus Break	Phase 2	Document	Tommy
MP-UCEFR	Use cases in extended format review	Verifies if the extended use case format is correct	5/6/22	Bonus Break	Phase 2	Document	Maria
DP-CM	Conceptual model	It is the class model without methods	5/6/22	Bonus Break	Phase 2	Document	Ainara
MP-CMR	Conceptual model Review	Verifies if the conceptual model is correct	5/6/22	Bonus Break	Phase 2	Document	Brian
DP-OC	Operation Contracts	Defined for every action in the actor column in the expanded case format to describe what happens when the actor	5/6/22	Bonus Break	Phase 2	Document	Elvis

		interacts with the system					
MP-OCR	Operation Contracts Review	Verifies if the operation contracts are correct	5/6/22	Bonus Break	Phase 2	Document	Alvaro
DP-CD	Class diagram	Shows the connections between classes. Every class that appears in the sequence diagram appears in the class diagram.	5/6/22	Bonus Break	Phase 2	Document	Justin
MP-CDR	Class diagram review	Verifies if the class diagram is correct	5/6/22	Bonus Break	Phase 2	Document	Tommy
DP-SD	Sequence diagrams	One for every operation contract to show when an actor and systems interacts with a each other	5/6/22	Bonus Break	Phase 2	Document	Maria
MP-SDR	Sequence diagrams review	Verifies if the sequence diagrams are correct	5/6/22	Bonus Break	Phase 2	Document	Ainara
DP-TSD	Transition states diagram	Used to show the performance of a class	5/6/22	Bonus Break	Phase 2	Document	Brian

MP-TSDR	Transition states diagram review	Verifies if the transition state diagrams are correct	5/6/22	Bonus Break	Phase 2	Document	Elvis

<u>Relationships</u>:

Dependency

Offer	Budget		March 20, 2022
Quality Plan	Software Configuration Management Plan	Estimation	March 20, 2022
Quality Plan	Software Configuration Management Plan	Feasibility Analysis	March 20, 2022
Use case model draft	Use case model	Use case model review	March 27, 2022
Use case model	Prioritization of use cases		March 27, 2022
Use case model	Definition of high-level use cases		March 27, 2022
Use case model	Estimation	Schedule	April 19, 2022
Use cases in extended format	Operation Contract		May 06, 2022
Class diagram	Sequence Diagram		May 06, 2022
Operation contract	Sequence Diagram		May 06, 2022

Derivation

Offer and Budget	Quality Plan	Software Configuration Management Plan	Estimation	March 20, 2022
Feasibility	Feasibility			March 27,

Analysis	analysis Review			2022
Prioritization of use cases	Prioritization of use cases review			March 27, 2022
Definition of high-level use Cases	Definition of high-level use Cases review			March 27, 2022
Estimation	Estimation review			April 19, 2022
Schedule	Schedule review			April 19, 2022
Use cases in extended format	Use cases in extended format review			May 06, 2022
Conceptual model	Conceptual model review			May 06, 2022
Operation Contracts	Operation Contracts review			May 06, 2022
Class diagram	Class diagram review			May 06, 2022
Sequence diagram	Sequence diagram review			May 06, 2022
Transition states diagram	Transition states diagram review			May 06, 2022
Definition of high-level use cases	Use cases in extended format			May 06, 2022
Use cases in extended format	Operation contracts			May 06, 2022
Operation contracts	Sequence diagram			May 06, 2022

Succession

CE Code	Previous Version	Next Version	Date
MP-O	N/A	1	February 20, 2022
MP-O	1	2	February 25, 2022
MP-B	N/A	1	February 20, 2022
MP-B	1	2	February 25, 2022
CP-SCMP	N/A	1	March 01, 2022
CP-SCMP	1	2	March 07, 2022
DP-UCM	N/A	1	March 18, 2022
DP-UCM	1	2	March 25, 2022
DP-UCM	2	3	March 28, 2022
DP-PUC	N/A	1	March 25, 2022
DP-PUC	1	2	March 28, 2022
DP-HLUC	N/A	1	March 25, 2022
DP-HLUC	1	2	March 28, 2022
MP-E	N/A	1	April 10, 2022
MP-E	1	2	April 19, 2022
MP-S	N/A	1	April 10, 2022
MP-S	1	2	April 19, 2022
DP-UCEF	N/A	1	April 26, 2022
DP-UCEF	1	2	May 06, 2022

DP-OC	N/A	1	April 26, 2022
DP-OC	1	2	May 06, 2022
DP-SDR	N/A	1	April 27, 2022
DP-SDR	1	2	May 06, 2022
DP-TSDR	N/A	1	April 27, 2022
DP-TSDR	1	2	May 06, 2022

For the class diagram and the use case model, the relationship would be a dependence relationship. This is because if something within one of these changes, it will directly affect the other. When one is updated, the other should be as well.

Using the quality plan and the software configuration management plan together, we are able to define the requirements in order to deduce the feasibility of our project. The feasibility analysis depends on these two (quality plan and software configuration management plan).

The use case model review depends on the use case model, which depends on the use case model draft. It all begins with the first use case model draft, which we use as a reference at the start. It later then gets refined and more polished to become the final version of the use case model and lastly will have the review of this model. Our final use case model then is used as a reference for our definition of high-level use cases. Our definition of high-level use cases depends on our final form of the use case model.

Both the prioritization of use cases and the definition of high-level use cases depend on the use case model to base the prioritization and the description of the use cases within our project.

The schedule takes into consideration the use case model and the estimation in order to make a proper schedule fit for the project at hand. It has to take into account certain elements from these two; therefore, it depends on what comes before it in order to make a good schedule to follow.

For the offer and budget, the relationship would be a dependency relationship, as they both directly affect one another and take into consideration a lot of similar factors. These two would then have a derivation relationship with the quality plan, estimation, and software configuration management plan. These three would then have a dependency relationship with each other because they all would affect the other. The software configuration management plan would ensure everything that needs to be controlled is being taken into account, while the quality plan would ensure that everything in terms of quality standards, practices, and resources are all up to par with

everything happening in the project, and the estimation. Lastly, the estimation would take all of these factors into account and would then provide a numerical quantity depending on everything that was analyzed previously.
In order to complete the operation contracts, we needed to use information from the use cases in extended format. Therefore, these contracts depend on these extended use cases and vice versa–if one changes, it directly affects the other. Additionally, the sequence diagram is also directly related to the class diagram. In order to complete the sequence diagram, we need to continuously reference the class diagram and if one changes, the other changes with it. Similar to this, the sequence diagram also depends on the completion of the operation contracts. So if the contracts change, the sequence diagram does too and they both directly correlate with each other.

The feasibility analysis review comes after the feasibility analysis, so the review is derived from the original feasibility analysis. Similarly, this is the case when you are dealing with the prioritization of use cases and the prioritization of use cases review, the definition of high-level use and definition of high-level use review, estimation and estimation review, the schedule and schedule review, use cases in extended format and use cases in extended format review, conceptual model and conceptual model review, operation contracts and operation contracts review, class diagram and class diagram review, sequence diagrams and sequence diagrams review, and transition states diagram and transition states diagram review. All of these reviews depend on the configuration element it's respectively reviewing. The review is derived from the original configuration element in order to come up with a finalized review. In essence, for those configuration elements that have a review CE associated with them, there is a derivation relationship between both, as the revisions will take place after they are defined. Additionally, the use cases in extended format are directly derived from the definition of high-level use cases, since it needs that information to become more indepth with extended amounts of information. Similar to the operation contracts being derived from the definition of high-level use cases because we use the information found in the use cases to build and make out contracts. Lastly, our operation contracts then are used to directly make the sequence diagram–hence, the diagram is derived from the contracts we make.

The succession relationships are configuration elements in which we initially started working on, but then later edited in order to update these respective elements. The elements we considered succession are:

> Offer, Budget, Software configuration management plan, use case model, Prioritization of use cases, Definition of high-level use cases, Definition of high-level use cases, Estimation, Schedule, Use cases in extended format, Operation contracts, Sequence diagrams, and Transition states diagram.

4.11 Configuration auditing

Continues in section 10.

5. Quality Plan

CONTENT OF THE QUALITY ASSURANCE PLAN FOR THE INFORMATION SYSTEM

In the successive points of the document, the detailed tasks that are going to be carried out in the fulfillment of the Quality Assurance Plan will be exposed to check that the whole project fulfills the necessary quality criteria and that they have been considered as indispensable for the correct accomplishment of the project.

The revisions will be made as the project phases are completed until the final and complete design of the product is reached.

Those responsible for carrying out the revisions and accepting the validity of the products will be Elvis Vasquez as Quality Manager and Brian Jesse Gatukui Kimani as Project Manager. In addition, all the members of the work team must carry out the revisions assigned by the Project Manager and communicate to the two people in charge of the Quality Assurance Plan in the event that any fault is found.

The following points of the document detail the specific reviews that will have to be carried out in compliance with the Quality Assurance Plan. The establishment of this quality assurance plan will begin in the System Feasibility Study and will be applied throughout the development of the software project (analysis, design, implementation...).

For each of the revisions, an Audit Report must be added that includes the approval or rejection of the revised product, indicating, if necessary, the causes for rejection of said product.

REVIEW OF THE SYSTEM'S FEASIBILITY STUDY

DOCUMENT REVIEW

Elvis Vasquez, as Quality Manager, will confirm that the requirements have been specified in a structured way, with a precise and complete content, as established in the Quality Assurance Plan. Our Quality Manager will ensure that the requirements specification document offers the following features:

- Identification of absolutely all user requirements.
- Consistency between the content of the document and its objective.
- Each requirement describes the functionality that corresponds to it.
- Correspondence between the requirements of the document and the requirements obtained from the user, so the requirements specification is complete.
- Description of the requirements in clear, unambiguous language and therefore precise
- The feasibility study is self-descriptive, as its structure and content are described.
- A requirements traceability matrix shall be carried out to check that all user requirements have at least one software requirement associated with them and are thus present in the system design.

REVIEW OF USE CASES

REVISION OF THE USE CASE DIAGRAM

Use cases are a very important tool in the software development process and we use them to estimate activities before modeling or building a software development process.

With the use cases we have the functionalities and characteristics or basic requirements of the system. They are not based on any language so they are independent of them.

From the use cases, using the use case method, the size of the software will be estimated. The requirement to be able to use this tool is to define a use case model that represents well the domain of the problem to be addressed.

Maria Opland, as Quality Responsible, must carry out the revision of the Use Case Diagram, for this she must verify that the use case diagram complies with the following:

- The use case diagram describes the behavior of the system, i.e. the complete functionality of the software project to be developed.

- The use case diagram includes all identified use cases representing all system functionalities.

- The use case diagram includes all the actors identified and involved in the system.

- The use case diagram includes all the dependencies and relationships between actors and use cases.

- The use case diagram complies with the graphic notation defined in UML modeling language.

- The use case model includes a glossary of terms that describes the terminology used.

REVIEW OF HIGH-LEVEL USE CASES

Elvis Vasquez, as Quality Manager, must carry out the revision of the high level Use Cases, to do so, she must verify that they comply with the following

- The high-level use cases contain the name, actors, description and type of use case.

- Each use case describes how to achieve a single goal, that is, it describes a feature of the system.

- Each use case contains a textual description of the functionality associated with the appropriate level of detail, including ways in which the intended actors could work with the system. The description will use the language of the end user.

- The use cases do not describe internal system functionality, nor do they explain how it will be implemented. They do not include technical jargon.

- Each use case shows the steps that the actor follows to perform an operation.

- The use cases comply with the graphic notation defined in UML modeling language.

CONFIGURATION MANAGEMENT PLAN REVIEW

CONFIGURATION MANAGEMENT PLAN REVIEW

Elvis Vasquez, as Quality Manager, must carry out the revision of the Configuration Management Plan, to do so she must verify that it complies with the following:

- The project includes a Configuration Management Plan for the control and management of changes in which the activities to be carried out are established that allow the control and management of changes in the project.

- The Configuration Management Plan complies with IEEE Std. 828 - 2005: "IEEE *Standard for Software Configuration Management Plans*" and ANSI/IEEE Std. 1042 - 1987: "*IEEE Guide to Software Configuration Management*".

- The management of the configuration defined in the SCM is carried out during all phases of the software project development, including maintenance and change control.

- The SCM describes a change and version control mechanism that ensures the production of quality software.

- The MTS includes the procedure for generating the necessary documentation for recording and monitoring the changes that occur during the development of the project.

REVIEW OF PROJECT ESTIMATION AND PLANNING

REVISION OF ESTIMATE

When planning a project, an estimate of the cost and human effort required must be obtained. Estimation is one of the crucial activities in the software project management process, necessary for project planning.

Maria Opland, as Quality Responsible, must make the revision of the estimate made for the software development project, for this she must review the following:

- The method used to estimate the effort for the development of the software project uses size-oriented metrics based on points of use cases.

- Before each iteration, verify that the estimate has been made taking into account the use cases included in the estimate.

- The use case points for each of the iterations have been calculated following the procedure established for this estimation method which includes the following steps:

    - Classify each iteration between actor and chaos of use according to its complexity and assign a weight according to it.

    - Calculate the complexity of each use case according to the number of transactions or steps in the case.

    - Calculate the Unadjusted Use Case Points of the iteration.

    - Calculate technical and environmental complexity factors.

    - Calculate Adjusted Use Case Points.

- Once the use case points have been obtained for an iteration, verify that the corresponding effort required to carry them out in that iteration has been calculated from them.

PLANNING REVIEW

Planning is the process of establishing objectives and choosing the means to achieve them. It is essential to carry out an analysis of the project in order to foresee from the beginning and during the development of the project the situations that may arise and to create the necessary conditions to be able to solve them or minimize the consequences that they may have on the development of the project and the achievement of the objectives.

Elvis Vasquez, as Quality Manager, must carry out the revision of the planning made for the software development project, for this she must verify the following:

- A prioritization of use cases to be developed has been carried out and the iterations that will make up the complete development of the software and the use cases included in each of them have been defined.

- An estimation of each iteration has been made based on Use Cases. Based on this estimate, planning will be carried out.

- Before starting an iteration, a planning of the iteration will be done based on the estimation of the effort needed according to the points of use cases.

- The planned planning for the development of the software project will be adapted and updated as the project progresses.

- Planning includes how many people should participate in the project team, what technical skills are needed, when to increase the number of people and who will participate.

- The planning done defines how the team that will work on the software development project will be organized.

- The planning follows the methodology applied to the software development project which is, in this case, incremental iterative based on use cases.

- A Gantt chart is included, representing all the activities to be carried out throughout the project development period. The diagram connects the different activities based on their relationships of precedence and defines the estimated resources and times for each activity.

- The Gantt chart reflects the tasks and key dates, the milestones and the dependency between tasks.

- The quality metrics to be applied to the planning carried out will be

- Speed at which objectives or requirements are completed in each iteration

- Urgency and priority of the completed requirements, to check if there is any misalignment with the project objectives and the organization's strategy.

- Requirements completed in iteration.

- Built-in changes and added requirements on the initial scope of iteration

- Number of requirements completed out of total requirements.

- Deviation of project results from initial planning

- Budget available, budget spent and financial deviation from initial planning.

- Customer satisfaction with regard to the results obtained.

TEST PLAN REVIEW

Elvis Vasquez, as Quality Manager, must carry out the revision of the Test Plan, for this she must do the following:

- It should be checked that there are rules for carrying out the tests so that it is possible to verify that these tests have been carried out, as well as indicating how to act in the event of differences between the expected result and the result obtained.

- A traceability matrix must be carried out to ensure that there is evidence to verify all software requirements.

REVIEW OF THE PRODUCTS OF THE ANALYSIS PROCESS

REVIEW OF USE CASES IN EXPANDED FORMAT

Maria Opland, as Quality Responsible, must carry out the revision of the Use Cases in expanded format, for this she must do the following:

- From each high-level use case, an expanded use case has been built, in each iteration.

- Each expanded use case is composed of two sections, the header that includes the name, actors, description and type of use case, and the body that describes typical events and alternatives to typical events.

- Expanded use cases define the initiator of the use case.

- The body of the use case consists of two columns describing the actions of the actor and the system responses to them.

REVIEW OF THE CONCEPTUAL MODEL OF THE ANALYSIS

Elvis Vasquez, as Quality Manager, must carry out the revision of the Conceptual Model, for this purpose the following must be verified:

- The analysis model represents the aspects of the problem in a way that is close to the concepts of the problem domain and describes the main characteristics of the system. The analysis model carried out in each of the iterations that make up the project will be validated.

- The conceptual model does not include implementation decisions. It will also be verified that it is independent of the implementation.

- The conceptual model complies with the graphic notation of the UML modeling language. You should also check that the notation has the necessary level of detail to represent the problem, without being overloaded.

- The conceptual model has been made through an object model or class diagram (without methods) that defines the system properties. The entities and the relationships between them have been identified for each iteration.

- The quality metrics to be applied to the conceptual model resulting from the analysis in each iteration are the following:

- Semantic quality: correspondence between the model and the domain, i.e. the model reflects the domain. The validity of the model will be verified, i.e. that all the facts included in the model are correct and relevant to the domain.

- Completeness: the model will be checked to ensure that all facts are correct and relevant to the domain.

- Language quality: the modeling language used to capture the domain is a language that is easy to understand by all participants. The formalization of the language allows the execution of the system.

- Syntactic quality: there is a correspondence between the externalization of the model and the extension of the language in which the model is written.

REVIEW OF OPERATING CONTRACTS

Elvis Vasquez, as Quality Manager, must carry out the revision of the operation contracts that are generated, for this purpose the following must be verified:

- For each case of use, there must be a contract of operation for each action of the actor.

- Each operating contract will consist of the following fields: name, responsibilities, cross references, notes, exceptions, output, preconditions and postconditions.

- Cross-references in the contract shall correspond to references to the requirements defined in the project that are resolved with the use case to which the operation contract belongs.

REVIEW OF THE DESIGN PROCESS PRODUCTS

CLASS DIAGRAM REVIEW

Assessing whether the design obtained meets the required quality level is important in order to know the effectiveness of the processes that have been modeled and whether or not they require great effort for their implementation.

Evaluating design class models by applying metrics allows for the detection of shortcomings and potential improvements from early stages of product development, preventing them from spreading to subsequent phases and enabling the creation of a robust system from its conception.

Maria Opland, as Quality Responsible, will have to carry out the revision of the Class Diagrams, for this she will have to check the following:

- Class diagrams will be made for each iteration with UML and the design will be totally independent of the implementation.

- The comprehensibility of the model or facility with which the class diagram can be understood, the analyzability of the model or facility offered by the class diagram to discover its deficiencies or errors, and the modifiability of the diagram or facility offered by the diagram to make a specified modification, either by error, by a concept not taken into account or by a change in requirements, shall be measured.

- The following metrics will be used to measure the structural complexity of the class diagrams:

    - Number of classes: total number of classes.

    - Number of attributes: total number of attributes.

    - Number of methods: total number of methods.

    - Number of partnerships: total number of partnerships.

    - Number of aggregations: total number of aggregation ratios.

    - Number of dependencies: total number of dependency relationships.

    - Number of generalizations: total number of generalization ratios.

    - Number of generalization hierarchies: total number of generalization hierarchies

- Number of aggregations: total number of aggregation ratios.
- WMC: class weighted methods, according to their complexity.
- Maximum ITL: is the maximum ITL value obtained for each class in a class diagram. For a class within a generalization hierarchy, it is the length of the longest path from the class to the root of the hierarchy.
- Maximum HAgg: is the maximum HAgg value obtained for each class in the class diagram. For a class within an aggregation hierarchy it is the length of the longest path from the class to the leaves.

- The proposed metrics are highly related both to maintenance time and to the comprehensibility, analyzability and modifiability of the designed class diagram.

REVIEW OF SEQUENCE DIAGRAMS

Elvis Vasquez, as Quality Manager, must carry out the revision of the sequence diagrams generated in the project during the design phase of each iteration, for this purpose the following must be verified:

- For each use case, sequence diagrams have been designed that define both the typical course and the atypical courses of the events defined in them.
- The sequence diagrams show the interaction represented by the sequence of messages between the class instances and actors. The diagrams show instances and events that describe the interaction between the classes.
- Time flows down the diagrams and shows the control flow from one participant to another.
- The UML notation is followed in the definition of the diagrams. The elements included in the sequence diagram are:
  - Name of the sequence diagram.
  - Lifelines for actors and class instances.
  - Messages between instances that define the method that the message calls on the receiving lifeline. In addition, the receiving line is linked to an interface or class.
  - Loops indicate the number of times the loop is executed if known.

REVIEW OF STATE DIAGRAMS

Elvis Vasquez, as Quality Manager, must carry out the revision of the state diagrams generated in the project during the design phase of each iteration, for this purpose the following must be verified:

- The defined state diagrams describe the behavior of the system, with each diagram showing the behavior of a single object during its entire life cycle.

- State diagrams contain states and transitions, and the transitions between them include the corresponding events or actions.

- The state diagram shows all possible states that the object goes through during its life in the application as a result of the events that reach it.

- There is an initial state and a final state and all states represented in the diagram are accessible.

6. Estimation

The most notable takeaway from the estimation excel sheet is that we have underestimated the amount of time that the project would take in the initial offer. We estimated a total of 6 months for BonusBreak's completion, yet, the excel estimation sheet revealed that given our priorities, we should expect 9.5 months until completion of our project. This is due to making specifications in the Technical Complexity Factors, where the estimation team decided to prioritize factors such as concurrency, security, end-user online efficiency, response time, and distributed systems. All the Technical Complexity Factor points totalled 43.5 points and gave us a Technical complexity factor of 1.035. These factors, multiplied with our 131 unadjusted use case points and 0.7925 Environmental Factor gave us a 107.45 total Adjusted Use Case Points. Multiplying these Adjusted Use Case Points with the 20 hours/use case determined, gives us that our project total coding time is 2149 man hours, which multiplying that by 100 and dividing by 40 hours totals us 5372.5 total man hours, which adjusted for total months for the project is 9.5 months.

Given the estimation, we should expect to have 3 workers for any phase of the project, as we also estimated in the budget; and an additional one for half of the total duration, which is the only difference regarding the people working in the project.

The amount of money needed to pay the salaries of the employees sums up to 85.625,10€, doubling the estimation we made during the budget due to the additional worker and the additional 3.5 months.

7. Planning

The Gantt chart that we developed in Microsoft Project is delivered as a separate file. This follows the time estimation that was done using the Excel sheet for estimation. We have decided that the project starts on February 25th and since we estimated that it would take 9 and a half months, it will end on November 18th. Phase 0 is estimated to last from February 25th to March 17th, phase 1 is estimated to last from March 17th to April 6th and phase 2 is estimated to last from April 6th to November 18th. Phase 2 is an iterative phase, which will have n iterations until the end of the project. The most time consuming tasks of the project are coding and testing, which will take about 55% of the time and resources available.

8. Planning and requirements specification

8.1 Feasibility study

The purpose of BonusBreak is to encourage employees at tech companies to have a better work-life balance and reduce burnout in the workspace by encouraging employees to take appropriate breaks, interact with their coworkers,enter their time working, and share their emotions while working. Participating in these activities will allow the employees to gain points, which they can redeem for points for monetary rewards or social events with coworkers. This will help achieve the second main goal of BonusBreak, which is to help workers feel properly compensated for their work by allowing them to redeem these points for rewards. All in all, BonusBreak aims to reduce burnout in tech company employees by encouraging them to take breaks, maintain good relationships with their colleagues, track their feelings at work, and to receive proper compensation for their work.

IDENTIFICATION OF STAKEHOLDERS IN THE SYSTEM

The stakeholders of our system are project managers of tech companies and the programmers that work for these companies.

8.1.1 Requirements definition

The requirements are going to be described as follow:

Identifier:	
Name:	
Priority:	**Source:**
Necessity:	
Clarity:	**Verifiability:**

Stability:
Description:

Figure 9: Requirements specification template

- The identification of the requirements will be done in the following way:
  - Identifier: UG-Snnn, where
  - U: indicates that this is a user requirement
  - G: General Requirement
  - S: admits the values:
    - C: Capacity requirement
    - A: Restriction requirement
  - nnn: Consecutive numbers to identify a requirement
- The name field summarizes the requirement
- The priority will have one of the following values:
  - High
  - Medium
  - Low
- The source field can have one of the following values:
  - Customer
  - Analysts
- The necessity field will have one of the following values:
  - High
  - Medium
  - Low
- The clarity field will be assigned one of the following values:
  - High
  - Medium
  - Low
- The verifiability field can have one of the following values:
  - High
  - Medium
  - Low
- Stability describes the duration of the requirement over the life of the software.
- The description field serves to explain the requirement.

FUNCTIONAL REQUIREMENTS

Identifier: **U-C001**
Name: **A user should be able to add the breaks and time working they have had during the day into the app**

Priority: High	**Source:** Customer
Necessity: High	
Clarity: High	**Verifiability:** Low
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The user needs to be able to add their breaks and their time working into the app in order to calculate points	

Identifier: U-C002	
Name: A user should answer a brief survey during sign up	
Priority: High	**Source:** Customer
Necessity: High	
Clarity: High	**Verifiability:** Low
Stability: The requirement will only be relevant when signing up for the app	
Description: The survey will include their current workstyle, means of de-stressing, interests outside of work and any current causes of stress, anxiety and unrest in the workplace.	

Identifier: U-C003	
Name: A user should be given random tasks to complete during the workday	
Priority: Medium	**Source:** Customer
Necessity: Low	
Clarity: High	**Verifiability:** Medium
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The user will be given a couple of small tasks during the day that they can complete for bonus points. The tasks will not be work related, and will mostly include a social element, so that the employee can take their mind off work for a moment.	

Identifier: G-C004	
Name: The system should be able to remind users to take scheduled breaks	
Priority: Medium	**Source:** Analyst
Necessity: Medium	
Clarity: High	**Verifiability:** Medium
Stability: The requirement will be relevant for the entirety of the software lifecycle	

Description: During the break the user will receive suggested activities to partake in. The activities will focus on a user's preferred interests and ways of de-stressing.

Identifier: U-C005	
Name: The user should be able to receive a notification to describe their current feelings/mood	
Priority: Low	**Source:** Customer
Necessity: Low	
Clarity: High	**Verifiability:** Medium
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The user will receive random notifications during the day to describe how they are feeling.	

Identifier: U-C006	
Name: The user should be able to see their teams total points in a graph	
Priority: High	**Source:** Customer
Necessity: Medium	
Clarity: Medium	**Verifiability:** Low
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The users should be able to see how many points their team has in comparison to the goal for the iteration. This should be visualized in some sort of a graph. Additional statistics could be included to indicate whether the team is on track to reach their goal or not.	

Identifier: U-C007	
Name: The user should be able to see their individual progress in a graph	
Priority: Medium	**Source:** Customer
Necessity: Medium	
Clarity: Low	**Verifiability:** Medium
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The user should be able to get a visualization of their progress in points and how implementing various recommended tasks affects their work style, productivity and improves overall wellness. A user is restricted to view other user's individual progress.	

Identifier: G-R008	
Name: The manager should be able to set a budget for the iteration reward	
Priority: Medium	**Source:** Management
Necessity: Medium	
Clarity: Medium	**Verifiability:** Low
Stability: This requirement is relevant at the start of each iteration	
Description: The manager of a team should be able to set a budget for the team reward in order to get appropriate reward suggestions from the team members. The budget can change from iteration to iteration according to how much the company wants to allocate.	

Identifier: U-C009	
Name: The user should receive bonus points for obtaining a streak	
Priority: Low	**Source:** Customer
Necessity: Low	
Clarity: High	**Verifiability:** Medium
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The user should be able to receive bonus points for obtaining a streak of using the app in consecutive days. The points received will be higher, when the streak is longer.	

Identifier: U-C010	
Name: The user should be able to submit suggestions for the iteration team reward	
Priority: Medium	**Source:** Customer
Necessity: Medium	
Clarity: Medium	**Verifiability:** Low
Stability: This will be relevant before an iteration, so that there is time to plan events	
Description: The employees should be able to submit suggestions for the team reward so that the reward can be something that the employees want. The cost of the reward suggestions should be within the budget that the manager has set.	

Identifier: U-C011	
Name: The user should be able to vote for the preferred reward	
Priority: Medium	**Source:** Customer

Necessity: Medium	
Clarity: High	**Verifiability:** Low
Stability: This will be relevant at the end an iteration, so that there is time to plan events	
Description: The user should be able to vote for the preferred reward that is suggested at the end of an iteration. This contributes to the employees being more motivated to accumulate points if the reward is attractive. The points accumulated throughout the iteration are used to vote.	

Identifier: G-R012	
Name: Users should be restricted from accessing other users individual progress	
Priority: High	**Source:** Analyst
Necessity: Medium	
Clarity: High	**Verifiability:** Low
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: Users should only be able to see their own individual progress, not others. This is a measure to prevent unnecessary pressure on employees to perform highly.	

Identifier: G-C013	
Name: The manager should be able to add and remove employees to/from their team	
Priority: High	**Source:** Management
Necessity: High	
Clarity: High	**Verifiability:** High
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The manager needs to be able to add employees to their team so that the users are members of the correct teams. If an employee starts working for another team or quits their job, it is also necessary to be able to remove members from a team.	

Identifier: G-R014	
Name: The manager should be able to set the number of points needed	
Priority: Medium	**Source:** Management
Necessity: Medium	
Clarity: Medium	**Verifiability:** Low
Stability: This requirement is relevant at the start of each iteration	

Description: The manager of a team should be able to set the number of points that is needed for the team to get the reward.

Identifier: U-C015	
Name: The user should be able to update their interests and ways of de-stressing	
Priority: Low	**Source:** Customer
Necessity: Low	
Clarity: Medium	**Verifiability:** Medium
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The user should be able to update their information from the initial survey. This will help improve the individual guidance within the app.	

Identifier: U-C016	
Name: The user should be able to describe their current feelings/mood	
Priority: Low	**Source:** Customer
Necessity: Medium	
Clarity: High	**Verifiability:** Low
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The user should be able to describe their current feelings/mood. This data will contribute to how the various tasks and breaks affect their work style, productivity and improves overall wellness.	

Identifier: U-C017	
Name: The user should be able to confirm that they have done a daily task	
Priority: Medium	**Source:** Customer
Necessity: Medium	
Clarity: Medium	**Verifiability:** Medium
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The user should be able to confirm to the system that they have done a daily task. This contributes to the points.	

Identifier: U-C018

Name: The manager should be able to approve/reject reward suggestions	
Priority: Low	**Source:** Management
Necessity: Low	
Clarity: High	**Verifiability:** Low
Stability: The requirement will be relevant at the start of each iteration	
Description: The team manager should be able to approve and reject reward suggestions from the team members. Reasons for rejecting might be that the suggestions are inappropriate or out of budget.	

Identifier: U-C019	
Name: The user should be able to open a user profile	
Priority: High	**Source:** Customer
Necessity: High	
Clarity: Medium	**Verifiability:** Medium
Stability: The requirement will be relevant for the entirety of the software lifecycle	
Description: The user should be able to see their user profile when they want.	

NON-FUNCTIONAL REQUIREMENTS

Identifier: G-C020	
Name: The system shall be compatible with the latest version available of Android OS and iOS.	
Priority: High	**Source:** Analysts
Necessity: High	
Clarity: High	**Verifiability:** High
Stability: The requirement is relevant for the entirety of the software lifecycle.	
Description: The system shall guarantee support for devices with Android OS version 6.0 and iOS version 10. The system also needs to be able to support newer and future versions of these operating systems.	

Identifier: G-C021

Name: The system shall be protected against different types of cyber attacks.	
Priority: Medium	**Source:** Analysts
Necessity: High	
Clarity: Medium	**Verifiability:** Low
Stability: The requirement is relevant for the entirety of its lifecycle.	
Description: The system shall be protected against different types of cyber attacks, such as SQL injection, DDOS, Man-in-the-Middle attacks.	

Identifier: G-C022	
Name: The system shall be available for use in multiple languages	
Priority: low	**Source:** Analysts
Necessity: medium	
Clarity: high	**Verifiability:** High
Stability: The application will have multilingual functionality for the entirety of the software lifespan	
Description: The application will have capability to allow users to select their preferred language and will first be developed in English and then over time we shall gradually add other languages to the app according to user demand	

Identifier: G-C023	
Name: The system must be ready for use during the working hours.	
Priority: High	**Source:** Client
Necessity: High	
Clarity: High	**Verifiability:** Medium
Stability: The requirement will be relevant for the entirety of the project's lifecycle.	
Description: The system shall guarantee its users a continuous use of the platform during working hours and proceed with maintenance operations out of that schedule.	

Identifier: G-R024	
Name: The system shall comply with the GDPR and protect the privacy of its users.	
Priority: High	**Source:** Analysts
Necessity: High	
Clarity: High	**Verifiability:** High

Stability: The requirement will be relevant for the entirety of the software's lifecycle.
Description: The system must comply with the General Data Protection Regulation of the European Union as well as the different privacy laws depending on the countries the system is used in. It will guarantee users' privacy protection.

Identifier: G-R025	
Name: Ethical Data Processing	
Priority: High	**Source:** Analysts
Necessity: High	
Clarity: Medium	**Verifiability:** Medium
Stability: The system should maintain data privacy throughout the Software lifetime.	
Description: The software will have checks in place to ensure ethical data processing. personal user data will not be used for financial gain or supplied to third parties for money. App functionality shall only rely on data that the user allows the software public access to and private information shall be stored in accordance with respective state privacy laws.	

8.1.2 Study of alternative solutions

The study of alternative solutions will be made with the following methodology:

- Firstly, we will specify a problem that is not considered in the requirements.
- Secondly, we will propose a solution to this problem that may deviate from the original idea of the project.
- Lastly, we will estimate the additional time and cost that would take to implement the solution.

The ideas given in the next section will be evaluated in section 8.1.4, stating why we think these solutions would be a good or bad addition to the project.

This methodology has been extracted from page 288 of SWEBOK v3.0, chapter 15-10.

8.1.3 Valuation of alternatives

Using the methodology from 8.1.2, we have defined the following alternatives:

A. Problem: users do not have an option to opt out of the team rewards.
   Solution: implement a functionality that lets users opt out of those rewards.
   Implementing this solution would take approximately ¼ of a month, as it would need an use case to be added in the Use case diagram.
B. Problem: users do not have a medical solution inside the app in case of already having the burnout syndrome.
   Solution: offering a functionality where users can communicate directly with psychologists and other medical staff to receive support from them and other ways to deal with burnout.
   Implementing this alternative would take around 4 to 6 use cases, so it would take around a month or month and a half to complete it.
C. Problem: Users do not have a chat feature.
   Solution: Put in place a chatbox feature for teams on the application that would allow teammates to interact.
   The implementation of this feature would also increase the timeline for building the app by around ½ - ¾ of a month due to 2-3 new use cases.
D. Problem: Users cannot view other teammates' progress for comparison purposes that may encourage taking of breaks.
   Solution: Add a team tracker feature that shows all team member contributions in order to encourage more participation.
   With approximately 2 new use cases, this feature would take half a month of time and resources to implement.

8.1.4 Solution selection

In this section will be detailed the result of analyzing the alternatives written in the previous section and detailing why they have been revoked or accepted.

A. This solution has been revoked in a first instance, as it would interfere with one of the main objectives of the project, which is encouraging employees to socialize in the workplace.
B. This solution would be a very good functionality to add to the project, as it would help users that are already suffering from burnout syndrome in a more meaningful or effective way, as they are being treated directly by experts in the field. The only restriction to adding this solution to the project would be that it would add around a month or month and a half of additional time in order to develop it, which would increase the cost of the project by around an additional 25%.
C. An issue that invalidates this solution is that BonusBreak hopes to increase social “in person” cohesion within the workplace while reducing burnout but the chat feature does not lead us to attaining this goal. There are also many other work chat applications like Slack therefore implementing this would not be unique and may be a waste of resources as it is not a crucial feature for BonusBreak. Therefore this solution must be revoked
D. A caveat to this might be that accessing teammate info on the app may be a catalyst to unhealthy competition or stigma for teammates who are not putting in more effort in accomplishing tasks. As a result we have chosen to revoke this option.

8.2 Use case model and traceability matrix

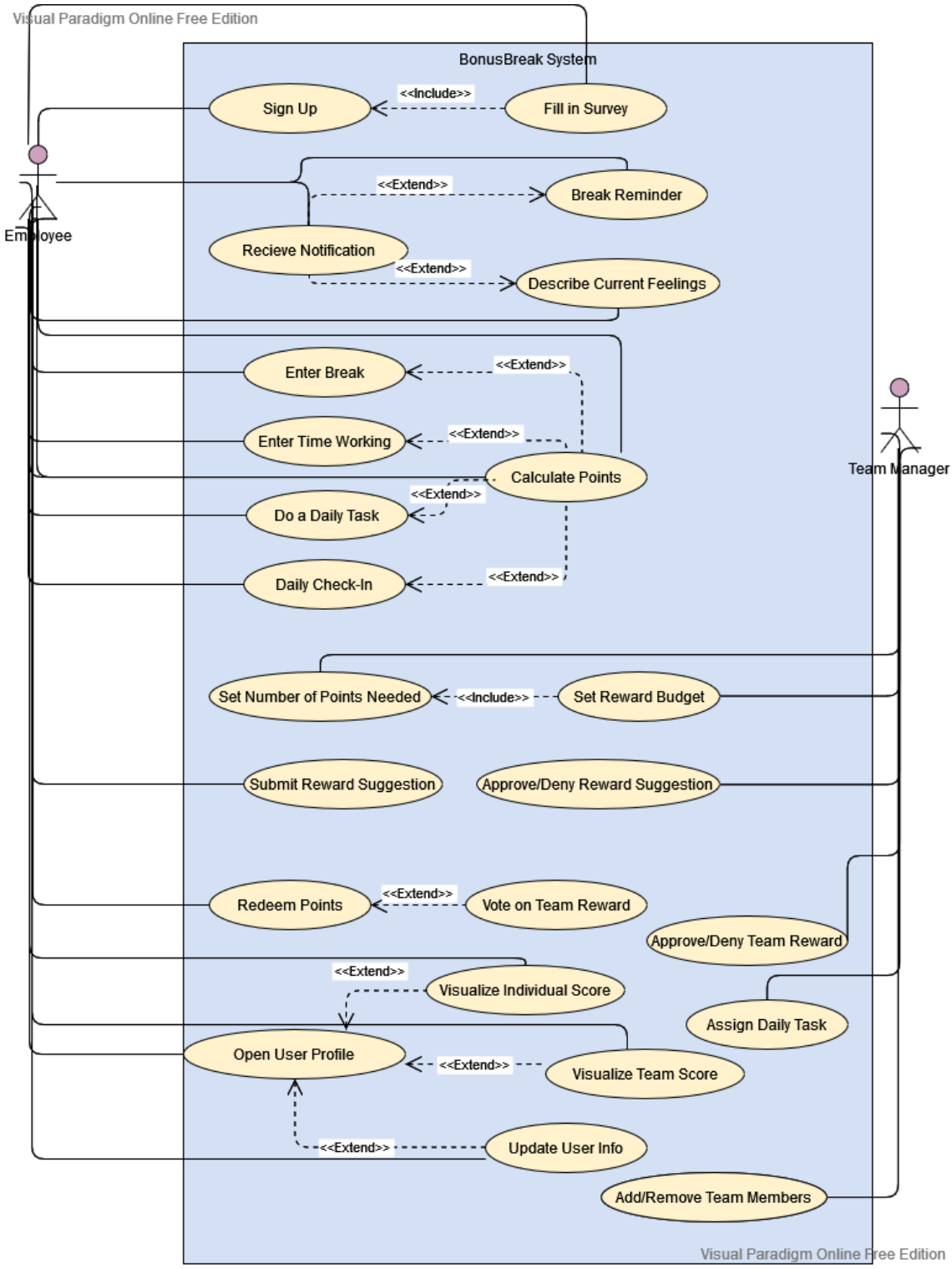


Figure 10: Use case diagram for BonusBreak

Use cases:

Code	Name
UC1	Sign up
UC2	Fill in survey
UC3	Break reminder
UC4	Receive notification
UC5	Describe current feelings
UC6	Enter break
UC7	Enter time working
UC8	Calculate points
UC9	Do a daily task
UC10	Daily check-in
UC11	Set reward budget
UC12	Submit reward suggestion
UC13	Approve/deny reward suggestion
UC14	Be given daily task
UC15	Vote on team reward
UC16	Open user profile
UC17	Visualize individual score
UC18	Visualize team score
UC19	Add/remove team members
UC20	Set number of points needed
UC21	Update user info
UC22	Approve/deny team reward
UC23	Redeem points

Figure 11: Table of use cases

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC 10	UC 11	UC 12	UC 13	UC 14	UC 15	UC16	UC17	UC18	UC19	UC20	UC21	UC22	UC23
U-C001						X	X																
U-C002	X	X																					
U-C003														X									
G-C004			X																				
U-C005				X																			
U-C006																		X					
U-C007																	X						
G-R008											X												
U-C009								X		X													
U-C010												X											
U-C011															X								X
G-R012																	X						
G-C013																			X				
G-R014																				X			
U-C015																					X		
U-C016					X																		
U-C017									X														
U-C018													X									X	
U-C019																X							

Figure 12: Traceability matrix

8.3 Use cases high level description

Use Case: Sign Up
Actors: Employee
Type: Primary, Essential
Description: A user opens the homepage of the app and clicks on a sign up button which will prompt the client for an email address, username, company they work for and contact information in order to create an account.

Use Case: Fill In Survey

Actors: Employee
Type: Primary, Essential
Description: Once the user has signed up, they will be prompted to answer a survey where they will detail work habits such as break habits, causes of stress or anxiety in the workplace, and sign in with their DevOps account to access their stories and sprint information to adapt the information gathered in the survey to their current work schedule. The information from this survey will be used to suggest break times and track improvement in feelings for the employee.

Use Case: Receive Notification
Actors: Employee
Type: Primary, actual
Description: The employee will receive a few notifications throughout the workday that present themselves as opportunities to gather points to reach a reward, but sparingly to not distract the employee too much from their work. An employee could receive a push notification reminding them when it's time to take a break and to describe how they're feeling today at work.

Use Case: Break Reminder
Actors: Employee
Type: Primary, Essential
Description: Throughout a given workday, BonusBreak will remind employees to take breaks in order to prevent burnout by overworking. These breaks serve as a method of relieving stress for employees in order for them to be more efficient as they are working throughout the given day/week. This will be one of the notifications the employees will receive.

Use Case: Describe Current Feelings
Actors: Employee
Type: Primary
Description: In order to track how employees are feeling as they are working, BonusBreak will send employees notifications in order for them to write down their feelings about the given day/situation. This will be one of the other notifications employees will receive throughout the workday.

Use Case: Enter Break
Actors: Employee
Type: Primary, Actual
Description: Users will have a trigger function/button to indicate to the app that they are entering a break period in order for them to keep a record of the breaks they have taken in the day and also trigger a timer for the break period.

Use Case: Enter Time Working
Actors: Employee
Type: Primary, Actual
Description: Similar to the "Enter Break" feature, Enter time working will allow the user to notify the app that they have finished their break period and are going back to

their working schedule. This will trigger points to be rewarded for the break taken as well as switch the timer from timing the break to timing the working hours for record keeping purposes.

Use Case: Do a Daily Task
Actors: Employee
Type: Secondary
Description: Every day, an employee will have optional tasks that are meant to improve their social and personal relationship with work. Tasks could be things such as complimenting a coworker, inviting a coworker to lunch, or even revising your schedule for the week and taking 10 minutes to organize yourself. These tasks are purely optional but reporting to BonusBreak that you participated in these will reward you bonus points.

Use Case: Daily Check-in
Actors: Employee
Type: Primary, Actual
Description: The employee will be able to check in on BonusBreak everyday in order to gain more points. This will be similar to describing their daily feelings, doing a daily task, or entering break/time working. This allows the system to reward the employee for checking in.

Use Case: Calculate Points
Actors: Employee
Type: Primary, Essential
Description: The points the Employee has accumulated from the activities Enter Break, Enter Time Working, Do a Daily Task, and Do a Daily Check-in are calculated by the system and displayed on the user's profile.

Use Case: Set Reward Budget
Actors: Team Manager
Type: Secondary
Description: The team manager will set a monetary budget (viewable by only them) for the designated reward. This monetary budget will help the manager to be able to translate this into how many points they deem appropriate for the reward and to be able to track if these funds are still available to carry out said reward.

Use Case: Submit Reward Suggestion
Actors: Employee
Type: Secondary
Description: The employee can write in a text box of 500 characters a suggestion for a reward to be evaluated and added by the Team Manager. This text is only visible to the manager and is not anonymous.

Use Case: Approve/Deny Reward Suggestion
Actors: Team Manager
Type: Secondary

Description: The team manager will be able to see the suggestion written by an employee and decide if this reward should be included for the employees by clicking an approve/deny button and adding the reward for the rest of the employees to see.

Use Case: Vote on Team Reward
Actors: Employee
Type: Secondary
Description: The employees within a given team will be able to choose on a reward they all agree upon, since all of their joined accumulated points are taken into consideration in order to be redeemed. This reward will serve as a way to further relieve stress as a team, instead of just on an individual level. This will only be possible by the team redeeming the points they have gained.

Use Case: Redeem Points
Actors: Employee
Type: Primary, Essential
Description: The employee is able to access the rewards section of the app and see the possible rewards set up by the Team Manager. Each reward will be valued a set number of points, and the employee must have exactly the number of points or over the amount needed to redeem a prize, if not, an error message will be displayed of insufficient points. If the reward is redeemed, the points will be deducted from their total accumulated points. Rewards can be individual rewards, where only the Employee's points are taken into consideration, or team rewards, where the whole team's total points are needed to be redeemed.

Use Case: Approve/Deny Team Reward
Actors: Team Manager
Type: Secondary
Description: The team manager will receive a notification that the team is trying to redeem a reward. It is up to the team manager to say if they are able to redeem this reward or if they need to change it to something else and deny the reward. If the reward is approved, the team members will be notified. If the reward is denied, the team members are reimbursed their points.

Use Case: Open User Profile
Actors: Employee
Type: Secondary
Description: The employee is able to view their user profile in order to access essential information such as their points accumulated as an individual, the points accumulated as a group, and edit their user information.

Use Case: Visualize individual score
Actors: Employee
Type: Primary, actual
Description: The employee is able to access the points they have accumulated individually by taking breaks, completing daily tasks, their time spent working, and by checking-in to the app.

Use Case: Visualize team score
Actors: Employee
Type: Primary, actual
Description: The employee is able to access the points they have accumulated as a group, where each team member's individual score is added up for a total team score, which can be used to redeem a team reward.

Use Case: Add/remove team members
Actors: Team Manager
Type: Primary
Description: The Team Manager is allowed to manage the employees inside a team by removing or adding employees to their designated team.

Use Case: Set number of points needed
Actors: Team Manager
Type: Secondary
Description: The team manager will be responsible for setting the points necessary for the reward. This will allow for teams to know what they are able to redeem points for, whether they want to continue saving their points or just choose an event to redeem for. This allows the manager to keep track of how much the company will spend if the team(s) continue to save up or redeem their points.

Use Case: Update user info
Actors: Employee
Type: Secondary
Description: The user can modify any information provided on the sign-up process, such as their email address, username, company they work for and contact information, and they are allowed to re-do their registration survey to change working times and breaks as needed.

Use Case: Be given daily task
Actors: Team Manager
Type: Secondary
Description: The Team Manager is allowed to create or modify custom daily tasks to give to specific or all employees depending on observations made around the workplace for how to improve employee's relationship with their job environment and peers. They are allowed to enter a string of a single sentence for a title, a paragraph for a description, and the number of points that are achievable by doing this task.

8.4 Use cases prioritization

In order to decide which usage cases are to be processed first, we need to sort them according to priority. The characteristics of a specific use case that will make a use case have a high priority are the following:

a. If it represents an important process in the line of business.
b. Whether it includes complex functions.
c. If it has a significant impact.

In order to carry out the classification, each case of use can be assigned a numerical valuation of each of these points, in order to obtain a total score by applying weights to each section. According to these criteria, values from 1 to 10 will be assigned to each use case and each criterion will be weighted according to the following:

Weighting	**0,5**	**0,25**	**0,25**	**Sum**	**Order**
Use case	**a**	**b**	**c**		
UC1	9	3	9	7.5	3
UC2	7	2	4	5	19
UC3	7	4	8	6.5	6
UC4	8	6	9	7.75	2
UC5	7	3	7	6	10
UC6	8	4	8	7	5
UC7	6	4	7	5.75	11
UC8	9	9	9	9	1
UC9	5	4	6	5	18
UC10	6	4	7	3.75	23
UC11	7	2	6	5.5	16
UC12	5	1	6	4.25	22
UC13	5	1	6	4.25	21
UC14	6	3	5	5	20
UC15	6	5	6	5.75	12
UC16	8	7	7	7.5	4
UC17	7	5	6	6.25	9
UC18	7	5	6	6.25	8

UC19	5	5	6	5.25	17
UC20	7	3	6	5.75	13
UC21	6	4	6	5.5	15
UC22	7	2	7	5.75	14
UC23	7	6	6	6.5	7

Based on the prioritizations above, we have come up with 3 main development cycles centered on:

1. Initial User sign up
2. Updating User info upon interaction with the platform
3. Calculating points/rewards after user-accomplished tasks.

FIRST CYCLE

- Sign Up
- Fill in survey
- Open user profile
- Set reward budget
- Set number of points needed
- Add/remove team members
- Visualize individual score
- Visualize team score
- Be given daily task

SECOND CYCLE

- Update user info
- Break reminder
- Receive notification
- Describe current feelings
- Enter break
- Enter time working
- Do a daily task
- Daily check-in

THIRD CYCLE

- Calculate points
- Submit reward suggestion
- Vote on team reward
- Approve/deny reward suggestion
- Approve/deny team reward
- Redeem points

9.Construction

9.1 First Iteration

9.1.1 First iteration analysis

Expanded format use cases description

- **Use Case: Sign Up**
- **Actors:** Employees & Managers
- **Purpose:** To create an account in order to be able to use BonusBreak
- **Overview:** An employee wants to join the BonusBreak team of their workplace. The first time they open the app they will get the options to sign in or sign up. As they don't have an account yet, they press the sign up button and start the process of signing up for BonusBreak. They follow the steps that appear on the screen and upon completing all the steps they will have successfully signed up.
- **Type:** Primary and essential
- **References:** *Functions:* U-C002
- **Typical course of events:**

Actor	**System**
1. This use case begins when an user presses the "Sign Up"-button	2. Show input form for information like name, email, workplace and password
3. Enter the required information	4. Create an account for the user in the database and redirect the user to the sign up survey

- **Alternative courses:**
- **Line 4:** One or more of the input fields contain invalid values, like for example too weak password or an email without an "@"-sign. An error message appears above the fields that are incorrect and the account is not created.

- **Use Case: Fill in survey**
- **Actors:** Employee
- **Purpose:** The user of BonusBreak fills in a survey when signing up, in order to tailor the experience of the app for each individual.
- **Overview:** When the user has signed up, they fill in a survey. The survey should include information about their current workstyle, means of de-stressing, interests outside of work and any current causes of stress, anxiety and unrest in the workplace.

- **Type:** Primary and essential
- **References:** *Functions:* U-C002
- **Typical course of events:**

Actor	System
1. This use case begins after a user has signed up for BonusBreak	2. It presents the survey as several questions
3. Fill in survey and click “Finish”	4. Process the survey to tailor the app

- **Alternative courses:**
- **Line 4:** Some of the questions are not answered. The system lets the user know which questions are not filled out and lets the user fill them out.

- **Use Case: Open user profile**
- **Actors:** Employee
- **Purpose:** The user should be able to view their profile at any time.
- **Overview:** The user should be able to view their profile whenever they want, when clicking the icon for “My profile”.
- **Type:** Secondary
- **References:** *Functions:* U-C019
- **Typical course of events:**

Actor	System
1. This use case starts when the user click on the icon for “my profile”	2. It displays the users profile

- **Alternative courses:**

- **Use Case: Set reward budget**
- **Actors:** Manager
- **Purpose:** Setting the monetary budget for the designated reward
- **Overview:** At the start of the iteration, the manager presses the “Set reward budget”-button and enters how much money that will be available for the iteration reward. This should correspond to the number of points needed.
- **Type:** Secondary
- **References:** *Functions:* G-R008
- **Typical course of events:**

Actor	System
1. This use case starts when the manager	2. Show slider with numerical values

uses their admin account to access the “Set reward budget”-panel	
3. Slide the slider to the desired reward budget	4. Update database with reward budget

- **Alternative courses:**

- **Use Case: Set the number of points needed**
- **Actors:** Manager
- **Purpose:** Setting the goal for the iteration that is used to determine whether the team receives the iteration reward
- **Overview:** At the start of an iteration, the manager of a team presses the “Set iteration goal”-button and enters the number of points required for the iteration. The goal will be visible to the entire team so that they know how many points are required to receive the iteration reward
- **Type:** Secondary
- **References:** *Functions:* G-R014
- **Typical course of events:**

Actor	**System**
1. This use case starts when the manager uses their admin account to access the “Set iteration goal”-panel	2. Show slider with numerical values
3. Slide the slider to the desired iteration goal	4. Update database with iteration goal

- **Alternative courses:**

- **Use Case: Add/remove team members**
- **Actors:** Manager
- **Purpose:** The manager should be able to add and remove members of the team, in order to be consistent with the real life workplace teams.
- **Overview:** The manager would want to add an employee to their team when there is a new hire, or someone has transferred teams within the company. Similarly, they would want to remove members from their team whenever an employee has left the real life workplace team.
- **Type:** Primary
- **References:** *Functions:* G-C013
- **Typical course of events:**

Actor	**System**

1. This use case starts when the manager uses their admin account to access the “Modify team”-panel	2. Shows the current members of the team, and a search bar for finding employees
3. The manager either searches for an employee and presses the “+”-icon to add them to their team, or presses the “x”-icon next to an existing member to remove them	4. Adds/removes member from the team, and updates the view with current team members

- **Alternative courses:**
- **Line 4:** If the employee does not have an BonusBreak account, the manager will not find them in the system and will not be able to add them to their team

- **Use Case: Visualize individual score**
- **Actors:** Employee
- **Purpose:** The employee should be able to access the points they have accumulated individually by taking breaks, completing daily tasks, their time spent working, and by checking-in to the app.
- **Overview:** The user should be able to access their individual score at all times, when clicking the icon for “Individual score”.
- **Type:** Primary and actual
- **References:** *Functions:* U-C007 and G-R012
- **Typical course of events:**

Actor	**System**
1. This use case starts when the user clicks on the icon for “Individual score”	2. It displays different statistics regarding individual score

- **Alternative courses:**

- **Use Case: Visualize team score**
- **Actors:** Employee
- **Purpose:** The employee should be able to access the points they have accumulated as a group, where each team member’s individual score is added up for a total team score, which can be used to redeem a team reward.
- **Overview:** The user should be able to access the team score at all times, when clicking the icon for “Team score”.
- **Type:** Primary and actual
- **References:** *Functions:* U-C006
- **Typical course of events:**

Actor	System
1. This use case starts when the user clicks on the icon for “Team score”	2. It displays different statistics regarding the team score

- **Alternative courses:**

- **Use Case: Be given daily task**
- **Actors:** Manager
- **Purpose:** The Team Manager should be able to create or modify custom daily tasks to give to specific or all employees.
- **Overview:** The manager should be able to create or modify custom daily tasks by clicking on the “Add daily task”- or “Modify daily task”-button. The task should contain a string of a single sentence for a title, a paragraph for a description, and the number of points that are achievable by doing this task.
- **Type:** Secondary
- **References:** *Functions:* U-C003
- **Typical course of events:**

Actor	System
1. This use case starts when the manager uses their admin account to access the “Add daily task”- or “Modify daily task”-button.	2. It displays the input fields: title, description, achievable points
3. The manager fills out all the input fields and clicks “Finish”	3. It adds the task to the database

- **Alternative courses:**
- **Line 4:** Some of the input fields are not answered. The system lets the user know which input fields are not filled out and lets the user fill them out.

Operation contracts

Action: 1. This use case begins when an user presses the “Sign Up”-button

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
signUp	Starts a session with the system so	Function: U-C002				No active sessions	A page has shown up with the sign up

	to show an sign up form	Use Case: sign up					form

Action: 3. Enter the required information

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
enterInfo (name: name, email: email, workplace: workplace, password: password)	Entering the information necessary to create a profile	Function: U-C002 Use Case: sign up		If values are invalid in fields, indicate that there has been an error	New user is created and stored in a database	No active sessions	A new user has been created and the user has been redirected to a personal survey

Action: 1. This use case begins after a user has signed up for BonusBreak

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
presentSurvey	Presents the user a survey in series of questions about their day to day	Function: U-C002 Use Case: Fill in survey				User has to be signed in	a page has shown up with the survey questions

Action: 3. Fill in survey and click “Finish”

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
fillSurvey (answers: answers)	Answer the questions presented and submit them	Function: U-C002 Use Case: Fill in survey		If some questions are not answered, lets the user know which questions need to be	Answers stored in database and sent to tailor	Questions have to be filled	Personal information is updated

				filled			

Action: 1. This use case starts when the user click on the icon for “my profile”

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
showProfile	User clicks the icon for “my profile”	Function: U-C019 Use Case: Open user profile				User has to be signed in	The user’s profile is displayed

Action: 1. This use case starts when the manager uses their admin account to access the “Set reward budget”-panel

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
accessRewardBudget	Manager accesses the“Set reward budget”-panel	Function: G-R008 Use Case: Set reward budget				Manager has to be signed in to admin account	A slider with numerical values is shown

Action: 3. Slide the slider to the desired reward budget

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
slideBudgetSlider (budget: budget)	Manager slides the slider to set the budget	Function: G-R008 Use Case: Set reward budget			Database is updated with new reward budget	Manager has to be signed in to admin account	Reward budget is updated

Action: 1. This use case starts when the manager uses their admin account to access the “Set iteration goal”-panel

Name	responsibilities	cross	notes	exceptions	output	preconditions	postconditions

		references					
accessIteratio nGoal	Manager accesses the“Set iteration goat”-panel	Function: G-R014 Use Case: Set the number of points needed				Manager has to be signed in to admin account	A slider with numerical values is shown

Action: 3. Slide the slider to the desired iteration goal

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
slidePointSlide r (points: points)	Manager slides the slider to set the points necessary for the goal	Function: G-R008 Use Case: Set the number of points needed			Database is updated with new iteration goal	Manager has to be signed in to admin account	Iteration goal points are updated

Action: 1. This use case starts when the manager uses their admin account to access the “Modify team”-panel

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
accessTeamM embers	Manager accesses the“Modify team”-panel	Function: G-C013 Use Case: Add/remov e team members				Manager has to be signed in to admin account	Shows a list of current members and a search bar for finding them

Action: 3. The manager either searches for an employee and presses the “+”-icon to add them to their team, or presses the “x”-icon next to an existing member to remove them

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions

AddRemoveMember (member: member, addOrRem: addOrRem)	Manager searches for team members and presses “+” or “x” icons to add/remove	Function: G-C013 Use Case: Add/remove team members		If the team member does not have a Bonus Break account the manager will not be able to find them to add or delete	Database is updated with new or deleted member	Manager has to be signed in to admin account	Shows the updated team view with new or deleted team member

Action: 1. This use case starts when the user clicks on the icon for “Team score”

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
accessTeamScore	User clicks on the icon for “Team score”	Function: U-C006 Use Case: Visualize team score				User has to be signed in	Shows a page with different statistics regarding the user’s team score

Action: 1. This use case starts when the manager uses their admin account to access the “Add daily task”- or “Modify daily task”-button.

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
accessDailyTask	Manager clicks “Add daily task”- or “Modify daily task”-button.	Function: U-C003 Use Case: Be given daily task				Manager has to be signed in to admin account	Displays input fields for modifying or creating a Daily task

Action: 3. The manager fills out all the input fields and clicks “Finish”

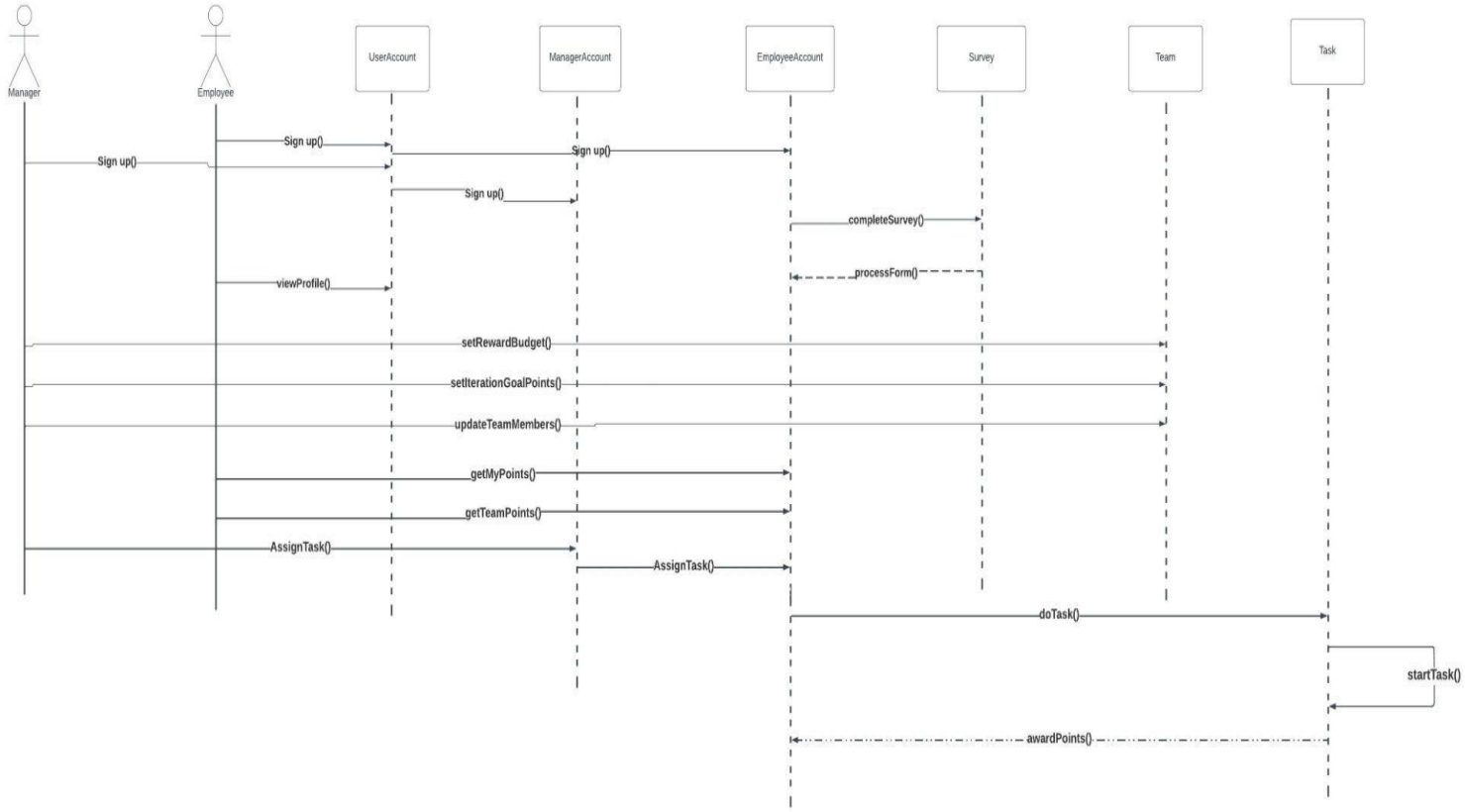
Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions

fillDailyTask (title: title, description: description, point: points)	Manager fills input field and clicks finish	Function: U-C003 Use Case: Be given daily task		If some fields are not answered, display error message	Database is updated with new Daily task	Manager has to be signed in to admin account	Daily tasks are updated

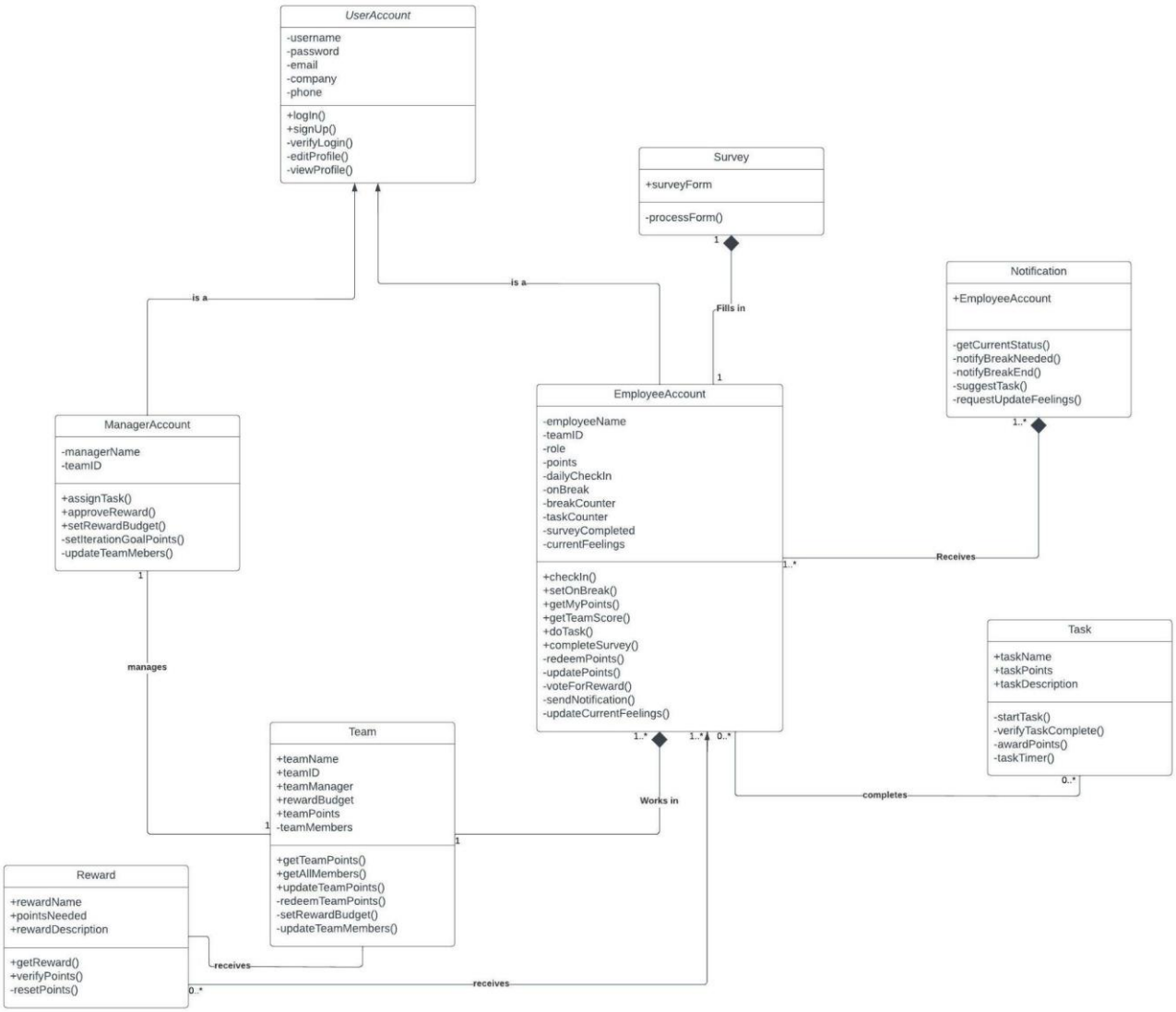
9.1.2 First iteration Design

Sequence diagrams

link to class diagram and first iteration sequence diagram on LUCID CHART

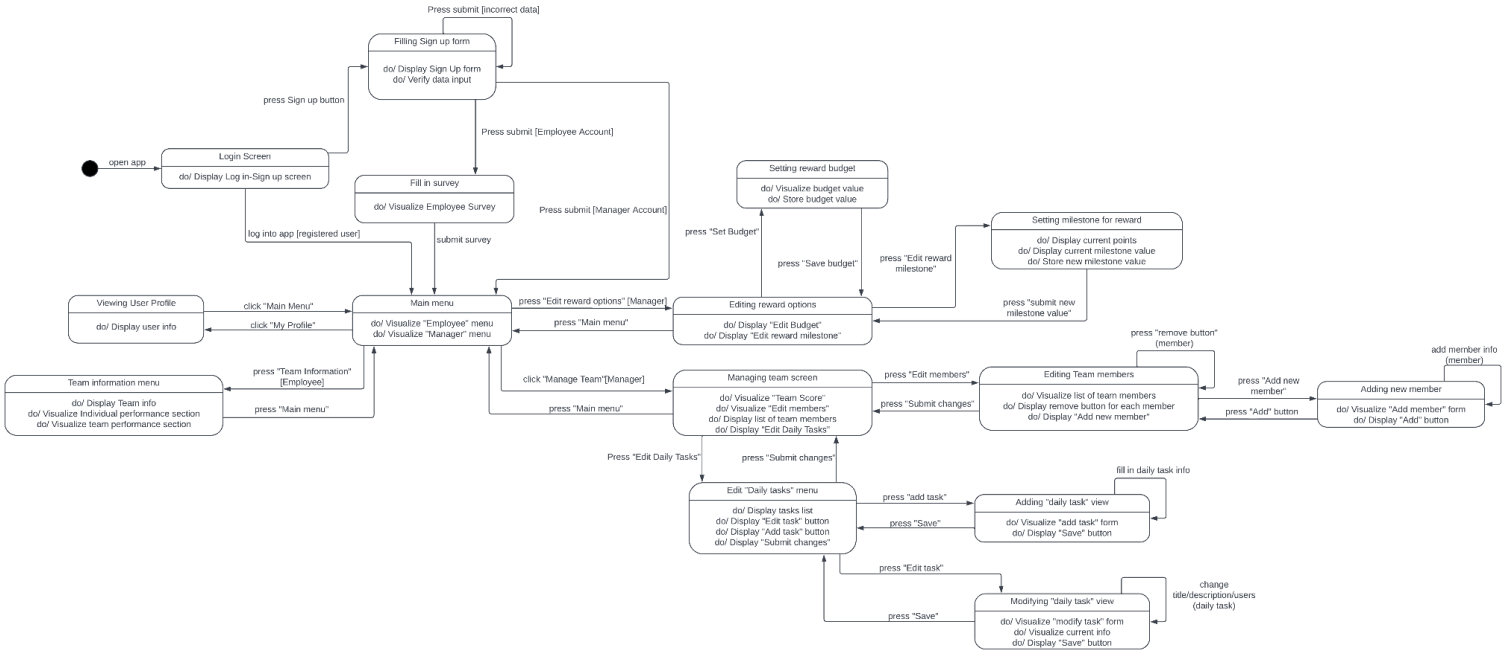

Manager
Employee
UserAccount
ManagerAccount
EmployeeAccount
Survey
Team
Task
Sign up()
Sign up()
Sign up()
Sign up()
Sign up()
completeSurvey()
processForm()
viewProfile()
setRewardBudget()
setIterationGoalPoints()
updateTeamMembers()
getMyPoints()
getTeamPoints()
AssignTask()
AssignTask()
doTask()
startTask()
awardPoints()

Class Diagram



Transition State Diagram

The Transition State diagram for the first iteration is the following one (if the size does not allow it to be visualized clearly, it can be seen here):



9.2 Second Iteration

9.2.1 Second iteration analysis

Expanded format use cases description

- **Use Case: Update user info**
- **Actors:** Employee
- **Purpose:** The user should be able to update their user information
- **Overview:** The user should be able to modify any information provided on the sign-up process, such as their email address, username, company they work for and contact information, and they are allowed to re-do their registration survey to change working times and breaks as needed.
- **Type:** Secondary
- **References:** *Functions:* U-C015
- **Typical course of events:**

Actor	System
1. This use case starts when the user clicks on the “Update user info”-button	2. It displays the different fields you can modify with the old information
3. The user modifies the preferred fields and clicks on the “Finish”-button	4. It updates the user information in the database

- **Alternative courses:**
- **Line 4:** Some of the input fields are not answered. The system lets the user know which input fields are not filled out and lets the user fill them out.

- **Line 4:** One or more of the input fields contain invalid values, like for example too weak password or an email without an "@"-sign. An error message appears above the fields that are incorrect and the account is not created.

- **Use Case: Break reminder**
- **Actors:** Employee
- **Purpose:** The employees should be urged to take breaks regularly in order to prevent burn out syndrome.
- **Overview:** During the workday the employee will receive a reminder through the app to take a break from work. The interval between the breaks will be decided by an algorithm that takes several factors into account.
- **Type:** Primary, Essential
- **References:** *Functions:* G-C004
- **Typical course of events:**

Actor	**System**
1. This use case gets triggered several times a day for an employee	2. Sends a message to an employee that says that it is time to take a break
3. Either accepts or declines the proposal of taking a break	4. The response from the employee is taken into account when deciding when to send the next break reminder

- **Alternative courses:**

- **Use Case: Receive Notification**
- **Actors:** Employee
- **Purpose:** In order for employees to be kept informed in real time, BonusBreak continuously sends notifications to employees about different tasks they should be doing throughout the day.
- **Overview:** Throughout the workday, BonusBreak presents different opportunities to gather points to reach a reward through sending these notifications to employees. Employees will receive push notifications reminding them of things such as taking a break or describing how they're feeling at work.
- **Type:** Primary, actual
- **References:** *Functions*: G-C004 and U-C005
- **Typical course of events:**

Actor	**System**
1. This use case begins after there are employees present	2. It presents notifications to each employee within the system
3. Check notifications and see if they have yet to complete any of the forms	4. Continue to push notifications if the employee has not completed tasks

- **Alternative courses:**

- **Use Case: Describe current feelings**
- **Actors:** Employee
- **Purpose:** For the employee to be reminded to describe their feelings.
- **Overview:** In order to track how employees are feeling as they are working, BonusBreak will send employees notifications in order for them to write down their feelings about the given day/situation. This will be one of the other notifications employees will receive throughout the workday.
- **Type:** Primary
- **References:** *Functions:* U-C016
- **Typical course of events:**

Actor	System
1. This use case starts when the user is registered in the system	2. It gives notification to each user in the system
3. The user receives notification and clicks on it	4. It displays a form for the user to fill out
5. The user fills out the form with their current feelings and clicks on the "Finish"-button	6. It store the information in a database

- **Alternative courses:**

- **Use Case: Enter break**
- **Actors:** Employee
- **Purpose:** The employee should be rewarded for taking regular breaks, and therefore needs to be able to tell the system when they have taken a break.
- **Overview:** After an employee has taken a break, they press the button for reporting a break in the app. They also input how long the break was. After reporting a break, the system will reward the employee with points, based on the length of the break and the amount of time since the last break.
- **Type:** Primary, Actual
- **References:** *Functions:* U-C001
- **Typical course of events:**

Actor	System
1. This use case starts when an employee presses the "Enter break"-button	2. Displays a slider for describing how long the break was and a "Confirm"-button

3. Slides the slider to the correct amount of minutes and presses the “Confirm”-button	4. Calculates an appropriate amount of points for the employee based on several factors and adds the points to the employee’s total

- **Alternative courses:**

- **Use Case: Enter time working**
- **Actors:** Employee
- **Purpose:** In addition to the amount of breaks, the amount of work an employee has worked contributes to the calculation of points. An appropriate balance between work and breaks leads to more points for the employee.
- **Overview:** At the end of a workday, the employee enters into the system how much time they have spent working that day. This leads to a calculation of points for the employee.
- **Type:** Primary, Actual
- **References:** *Functions:* U-C001
- **Typical course of events:**

Actor	**System**
1. This use case starts when an employee presses the “Enter working time”-button	2. Displays a slider for working time and a “Confirm”-button
3. Adjusts the slider to the correct amount of working time and presses the “Confirm”-button	4. Adds the information to the database, in addition to calculating the right amount of points and adding the points to the employee’s total.

- **Alternative courses:**

- **Use Case: Do a daily task**
- **Actors:** Employee
- **Purpose:** For the user to be able to perform daily tasks for bonus points.
- **Overview:** Every day, an employee will have optional tasks that are meant to improve their social and personal relationship with work. Tasks could be things such as complimenting a coworker, inviting a coworker to lunch, or even revising your schedule for the week and taking 10 minutes to organize yourself. These tasks are purely optional but reporting to BonusBreak that you participated in these will reward you bonus points.
- **Type:** Secondary
- **References:** *Functions:* U-C017
- **Typical course of events:**

Actor	System
1. This use case starts after the user clicks on the “Do daily task”-button	2. It displays the daily task
3. The user does the daily task and confirms to the system that it’s done	4. It calculates the amount of points the employee should receive and adds it to the employee’s total

- **Alternative courses:**
- **Line 4:** The user never does the daily task. Then the task gets removed and a new one will be given the next day.

- **Use Case: Daily check-in**
- **Actors:** Employee
- **Purpose:** For the employee to check in to the app daily
- **Overview:** The employee will be able to check in on BonusBreak everyday in order to gain more points. This will be similar to describing their daily feelings, doing a daily task, or entering break/time working. This allows the system to reward the employee for checking in.
- **Type:** Primary and actual
- **References:** *Functions:* U-C009
- **Typical course of events:**

Actor	System
1. This use case starts when the first time the user is accessing the app	2. It adds the appropriate amount of points to the employee’s total

- **Alternative courses:**

Operation contracts

Action: 1. This use case starts when the user clicks on the “Update user info”-button

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
presentUserFields	Presents the user the user information fields	Function: U-C015 Use Case: Update user info				User has to have a profile in the system and signed in	A page has shown up with the user information fields

Action: 3. The user modifies the preferred fields and clicks on the “Finish”-button

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
UpdateUserInfo(name: name, email: email, workplace: workplace, password: password)	Update user information if the user needs to change it	Function: U-C015 Use Case: Update user info		If values are invalid in fields, indicate that there has been an error	The user has been updated and it’s stored in a database	You have a profile in the system	The user information is updated and the user gets a notification on the screen indicating that the update has been successful

Action: 1. This use case gets triggered several times a day for an employee

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
breakReminder	Notification is sent to user to take a break	Function: G-C004 Use Case: Break reminder				User must be registered into the system	A message is sent to the employee to take a break

Action: 3. Either accepts or declines the proposal of taking a break

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
breakResponse(response: response)	User accepts or decline the break reminder	Function: G-C004 Use Case: Break reminder			Response taken into account when deciding when to	User must be registered into the system	A message goes away and decides when to send another reminder

					send the next one		

Action: 1. This use case begins after there are employees present

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
sendNotification	Notification is sent to user about tasks	Function: G-C004 & U-C005 Use Case: Receive Notification				Users must be registered into the system	A notification is presented to the employees about tasks

Action: 3. Check notifications and see if they have yet to complete any of the form

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
checkNotification	Checks if all tasks have been completed	Function: G-C004 & U-C005 Use Case: Receive Notification			If tasks all done update system to stop sending notifications	Users must be registered into the system	Sends reminder If tasks are still present

Action: 1. This use case starts when the user is registered in the system

Name	Responsibilities	Cross references	Notes	Exceptions	Output	Preconditions	Postconditions
receiveFeelingsNotificastion	Sends users notifications asking how they are feeling	Function: U-C016 Use Case: Describe current				Users must be registered into the system	The notification is displayed for the user

		feelings					

Action: 3. The user receives notification and clicks on it

Name	Responsibilities	Cross references	Notes	Exceptions	Output	Preconditions	Postconditions
displayFeelingsForm	Once user clicks the notification, the user is redirected to the feelings form	Function: U-C016 Use Case: Describe current feelings				Users must be registered into the system	A form is displayed for the user, in which they can describe their current feelings

Action: 6. The user fills out the form with their current feelings and clicks on the "Finish"-button

Name	Responsibilities	Cross references	Notes	Exceptions	Output	Preconditions	Postconditions
fillCurrentFeelingsForm(form: form)	User fills out the form with how they are feeling and click finish	Function: U-C016 Use Case: Describe current feelings			The information that the user provides is stored in a database	You have a profile in the system	redirected to a thank you for filling out page

Action: 1. This use case starts when an employee presses the "Enter break"-button

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
enterBreak	Employee clicks the"Enter Break"-button	Function: U-C001 Use Case: Enter break				User has to be signed in	A slider with numerical values and confirm button is shown

Action: 3. Slides the slider to the correct amount of minutes and presses the “Confirm”-button

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
slideBreakSlider (time: time)	employee slides the slider to set the time of break that was taken and confirms	Function: U-C001 Use Case: Enter break			Database is updated with points after break calculations are made	User has to be signed in	Points are updated with however much the break augmented it

Action: 1. This use case starts when an employee presses the “Enter working time”-button

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
enterWorkTime	Employee clicks the “Enter working time”-button	Function: U-C001 Use Case: Enter time working				User has to be signed in	A slider with numerical values and confirm button is shown

Action: 3. Adjusts the slider to the correct amount of working time and presses the “Confirm”-button

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
slideWorkSlider (time: time)	employee slides the slider to set the time of work that was done and confirms	Function: U-C001 Use Case: Enter time working			Database is updated with points after work calculations are	User has to be signed in	Points are updated with however much the work time augmented it

					made		

Action: 1. This use case starts after the user clicks on the “Do daily task”-button

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
enterDailyTask	Employee clicks the “Do daily task”-button	Function: U-C017 Use Case: Do a daily task	If the previous daily task has not been done, it will be replaced	If no daily task, say to come back the next day		User has to be signed in and manager has given a daily task	The daily task is displayed

Action: 3. The user does the daily task and confirms to the system that it’s done

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
completeDailyTask	employee confirms the daily task has been done	Function: U-C017 Use Case: Do a daily task			Database is updated with points after Daily task calculations are made	User has to be signed in and a daily task has to be present	Points are updated with however much the daily task points augmented it

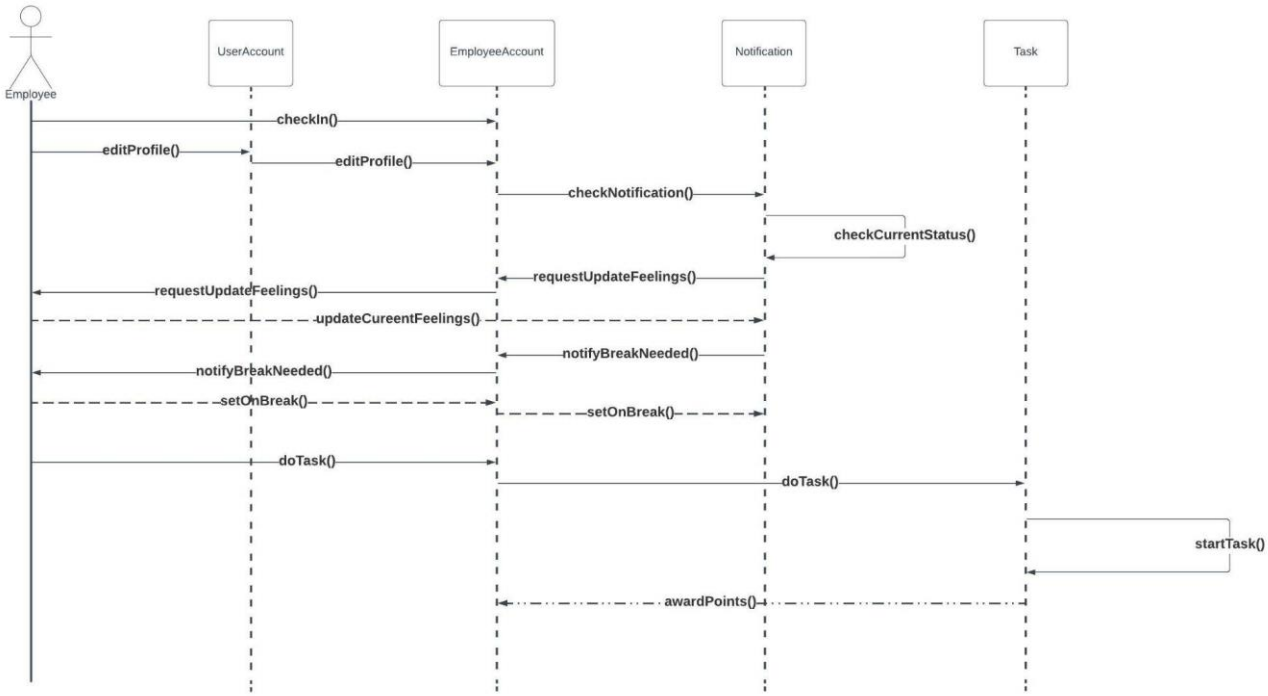
Action: 1. This use case starts when the first time the user is accessing the app

Name	responsibilities	cross references	notes	exceptions	output	preconditions	postconditions
dailyCheckIn	User opens application	Function: U-C009			Database is updated	User has to have a profile in the system	Points are updated with however much

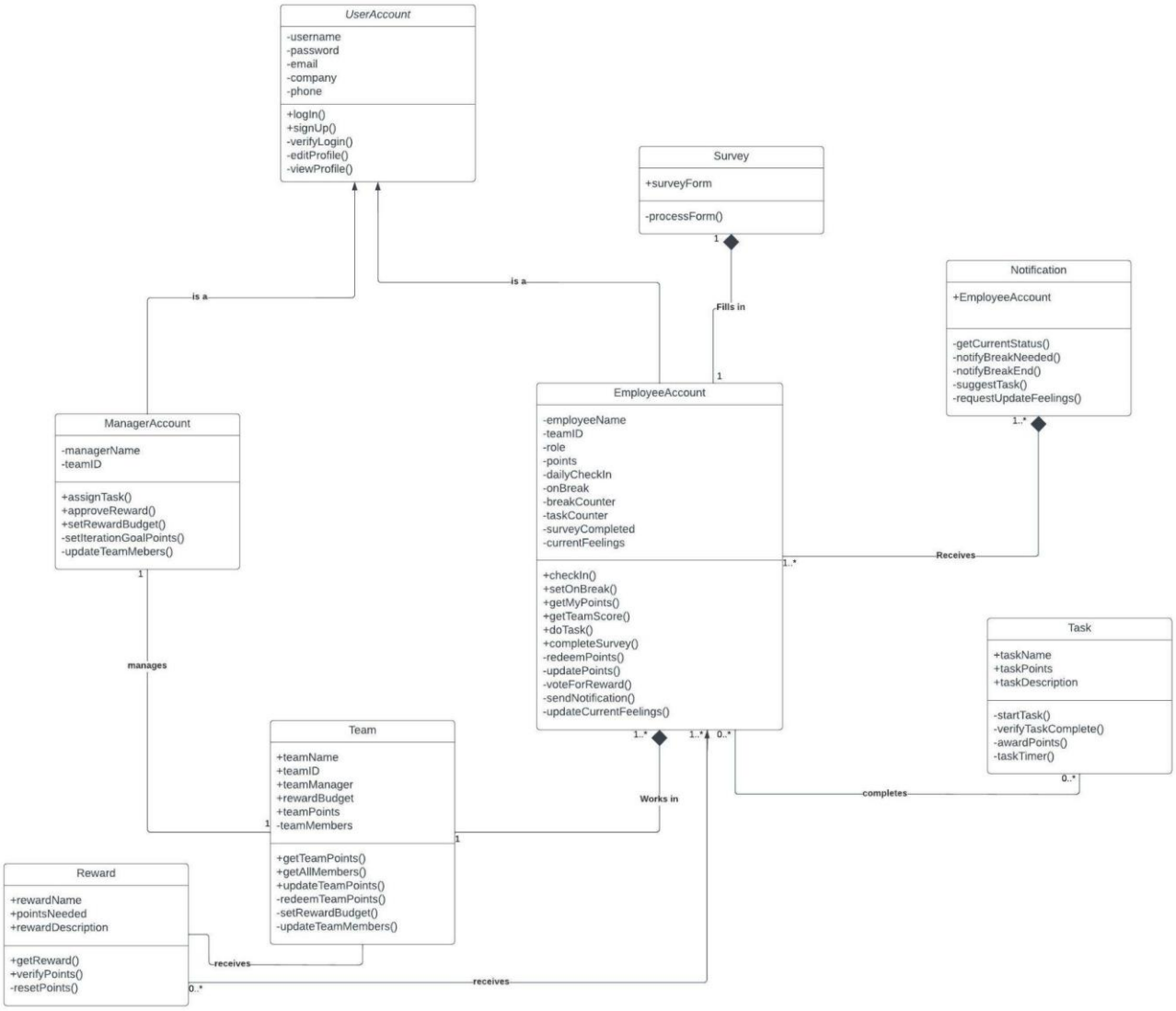
		Use Case: Daily check-in			with points after check in calculations are made	and signed in	the check in points augmented it

9.2.2 Second iteration Design

Sequence diagrams (Lucid Chart Link)

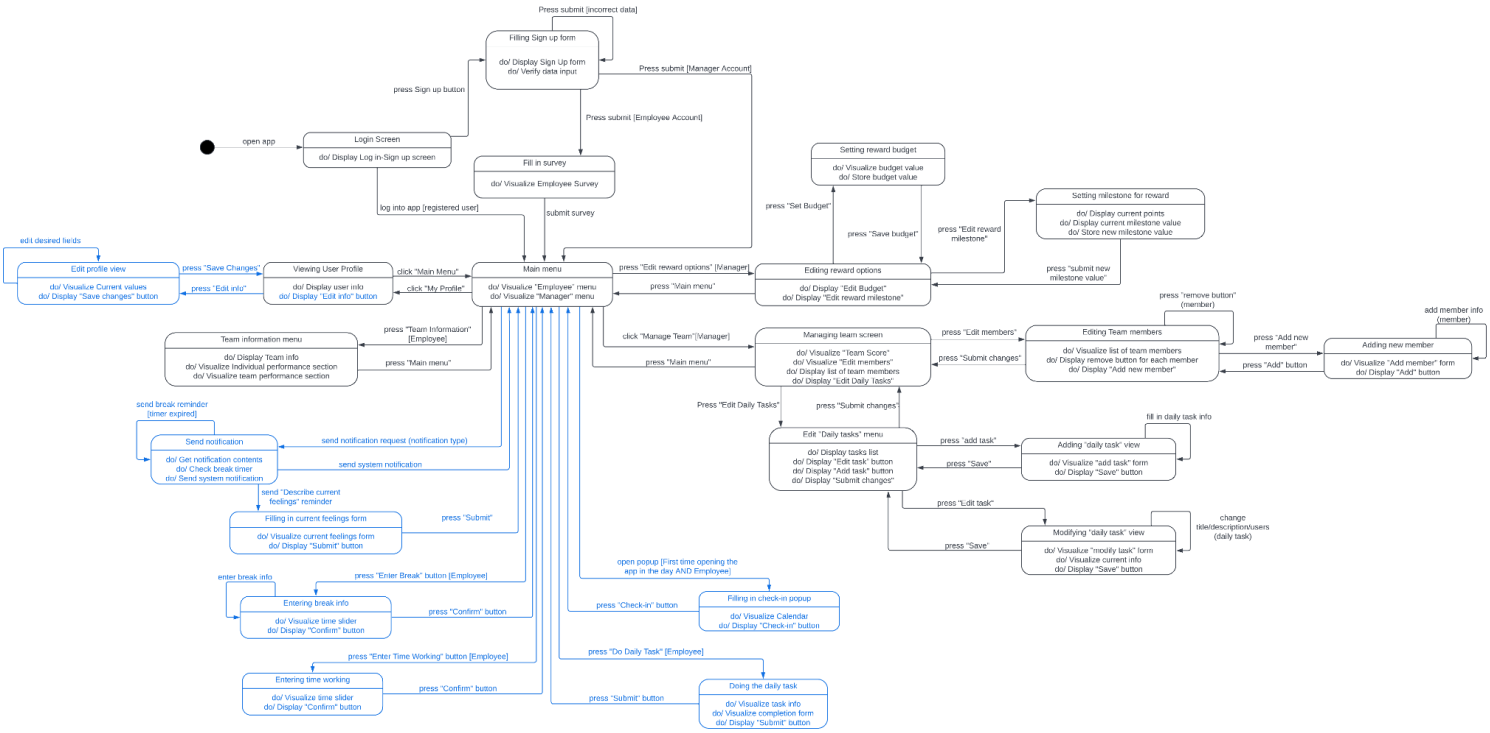


Class Diagram



Transition State Diagram

The state diagram for this second iteration is attached right below, corresponding to this second iteration only the states and transitions in blue, as it is an extension of the first iteration (if its size does not let the reader clearly take a look at it, it can also be found here):



10. Execution of the quality plan

After reviewing the quality plan, the offer and budget have all the requirements that are needed. The offer and budget were planned well. Using our use case model draft, we were able to determine factors within our offer and budget, which allowed us to validate and continue with our project. Our SCM plan is completed and has covered everything that has to be controlled within our project; therefore, our SCM plan review is also well since it describes the activities that need to be carried out throughout our project, allowing us to be in control of everything we need to control while also making sure that we can make any changes if needed. Lastly, our quality plan ensures that everything is up to par and that if anything that needs to be changed or revised, it can be done so that our project can be fulfilled correctly.

After reviewing the use case model draft, it fulfills all the requirements. The use case model then takes the draft and adds onto it. The complete model describes the behavior of our system while also identifying the use cases that represent all the system's functionality, actors within our system, and dependencies present in the system. It also complies with the modeling language used. The use case model review then verifies if the use case model is correct, which is it; thus, it fulfills the requirements.

After reviewing the software configuration management plan, it meets the requirements. It covers the control, management, maintenance of changes that will be

carried out throughout the project. It also includes a change and version control mechanism that allows us to oversee anything related to the quality software. The software configuration management plan then verifies if the plan is correct, which it is; therefore, it fulfills the requirements.

After reviewing the feasibility study, it meets the requirements since it is self-descriptive, since its structure and content are fully described within it. The feasibility analysis review then verifies the analysis which it properly does, ensuring that we are able to deduce the feasibility of the project; hence, it fulfills the requirements.

After reviewing the high-level description of the use cases, it contains the name, actors, description and type of the use cases involved. Each of the use cases involved contains a description of how it will achieve a certain objective/ feature of the system. It shows how the use cases are used as steps for the actor to perform an operation. Since it's high level, it does not describe the internal system functionality nor how it will be implemented, since the description will use language that the end user can interpret. The high-level description review then confirms that all of this is correct, which it does; thus, it fulfills all the requirements needed.

After reviewing the prioritization of use cases, it properly takes into consideration the use cases and their importance to rank them in priority order in order to see which will be processed first. After being weighed on certain criteria, the use cases then get ranked and put into cycles of how the use cases will be processed. After this, the prioritization of uses cases review, then reviews this prioritization to see if it is correct, which it does; hence, it fulfills all the requirements needed.

After reviewing the estimation, the method properly estimates the effort for the development of the software project based upon the points of the use cases. It takes into consideration the account of the use cases included in the estimate. It classifies each iteration between actor and the complexity, allowing an assignment of weight to it, calculates the complexity of the use cases according to the transactions in the respective use case, calculates the unadjusted use case points, the technical and environmental complexity factors and lastly also calculates the adjusted use case points. Then the estimation review obtains the use case points for every iteration and verifies if the corresponding effort required to accomplish them has been properly calculated for each respective use case. This is done properly; thus, it completes and fulfills all the requirements needed.

After reviewing the schedule, it properly takes into consideration the prioritization of uses cases that has been developed for the iterations that will complete the development of the software, the estimation of each iteration based on of the use cases, it is able to be properly updated and adapted as the project progresses, includes how many people will be participating in the project team and the technical skills needed, defines how the team will work on the software development and how it will be organized, follows the methodology properly, and includes a Gantt chart which properly represents all the activities that will be carried out throughout the project development period—the char connects activities based on their relationships and defines the

estimated resources and time for these given activities, reflects tasks and key dates, as well as the milestones and dependencies between tasks. The schedule review then properly checks all of these components of the schedule to ensure that it is all being carried out properly, which it is; therefore, these fulfill all the requirements.

After reviewing the use cases in extended format, it properly expands each high-level use case for each iteration. Each of these expanded use cases contains two sections: one is the header that includes the names, actors, the description and type of use cases, and the second is the body that describes the events associated with each use case. The body properly has two columns describing the actor and the system response to each. The use cases in extended format then ensures that all of these requirements have been properly met in order to ensure correctness.

After reviewing the conceptual model, the model represents the aspects of the problem in a way that describes the main characteristics of the system involved. This analysis will be carried out in each iteration of the project that will validate it. This model does not include any form of implementation decisions, since it is independent from the implementation itself. This conceptual model also follows UML modeling language and properly has enough level of detail, but not too much, in order to represent the problem at hand. This conceptual model has been made from a class diagram; thus, the entities and relationships present have been properly identified for each iteration. The conceptual model review then ensures that all quality metrics (such as the semantic quality, completeness, language quality, and syntactic quality) have all been properly met in order to ensure a model that is easy to understand while simultaneously providing all the information needed.

After reviewing the operation contracts, we properly have, for each use case, a contract of operation for every individual actor. Our operation contracts review ensures that every contract consists of all these fields: name, responsibilities, cross references, notes, exceptions, output, preconditions, and post conditions in order to deliver a proper contract. The review also ensures that the cross-references correspond to the requirements we defined in the project that are resolved with the use cases of which each operation contract pertains to.

After reviewing the class diagram, we have a class diagram for each iteration with UML and our design is independent of the implementation involved. We properly measure the comprehensibility of the model in order to make sure that it is easily understood, the analyzability of the model to allow for errors/deficiencies to be easily identified, and making sure that the modifiability is flexible in order to make any changes necessary to the diagram when necessary. In order to measure the structural complexity of our class diagrams, we use the following metrics: (total) number of classes, (total) number of attributes, (total) number of methods, (total) number of partnerships, (total) number of aggregation ratios, (total) number of dependency relations, (total) number of generalization ratios, (total) number of generalization hierarchies, (total) number of aggregations ratios, class weighted methods depending on the respective complexity, and the maximum ITL and maximum HAgg. Our class diagram review then properly ensures that all of these metrics are correct since all of these are related to the

maintenance time, comprehensibility, analysability, and modifiability of the designed class diagram. It assesses whether the design meets all of these required quality levels in order to make sure we know the effectiveness of the process that our model represents.

After reviewing the sequence diagrams, we have designed one for each use case that defines both the typical and atypical courses of the events defined in the respective case. Our diagram shows the interactions represented between the class instances and the actors involved. The diagram also shows instances and events that describe these interactions between class instances. It also includes the flowing of time within the diagram and shows the control flow of one participant to another. Our sequence diagram review then ensures that proper UML notation is followed within our diagrams. It also ensures that these following elements are present: name of the sequence diagram, lifelines for actors and class instances, messages between instances that define the method, and loops that indicate the number of times the loop is executed.

After reviewing the transition state diagram, it properly describes the behavior of the system with each diagram representing the behavior of each object throughout the project. The diagram contains states and transitions, and the transitions include the corresponding events/actions. Our state diagram review then ensures that it shows all possible states an object can go throughout the project due to possible interactions throughout the application. Lastly, it ensures that there's an initial and final state and that all states that are represented in the diagram are accessible.

11. Execution of the configuration management plan

All of this is done in section 4.10