REQUIREMENTS AND ANALYSIS DOCUMENT FOR COFFEEBREAK

VERSION: 1.1

DATE: THURSDAY, APRIL 6, 2017

AUTHOR: FELIX NORDÉN

This version overrides all previous versions.

1. INTRODUCTION

Today, where each individual gets more and more involved, the 24 hours a day we have at hand feels smaller and smaller. Being able to fit everything of interest into our day becomes more difficult and the time we can allocate is becoming smaller by the day.

Sleep deprivation has become a necessity and is a merit when applying for any form of profession. In conjunction, the demand on tools for planning has never been higher. They need to be fast, effective, and most importantly, they must have a small learning curve. However, amidst all of these time pressured schedules, the joy of life can be somewhat forgotten. Therefore, we in project group "Coffee Break" decided to create something that would fill the needs for efficient planning, and also give the user a reward to make planning a bit more fun.

Meet Caffeine Pill – the app that gets things done! It's a Todo-list application that will make planning a breeze and structuring your everyday life will never be easier!

1.1 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Todo-list – A list of different tasks that the user wishes to get done.

Task – The simple item that the user can add to his/her Todo-list

List task – A more complex version of a task, which creates the Task with the specified name and also contains a list of different tasks that the user can use to specify a list of tasks in a single place. This way, the user receives another option when they wish to organize/structure their Todo-list.

Time Category – The category which tasks can be sorted into which involve a certain timeframe

Label Category – Custom categories that the user can create through either adding custom labels/tags onto their task during creation, or through setting up static categories which are always visible.

Test-driven development – Before any new code is written for the application, a test for the specified component will be make using the different specifications for the component as guidelines. This will lessen the number of bugs in the end product. The procedure can be read in depth at https://blog.jetbrains.com/idea/2016/12/live-webinar-the-three-laws-of-tdd/ .

MVC – "Model, View, Controller", a design model used in the most applications/programs today. The Model is the database which handles all of the logic and calculations. The View is the what is

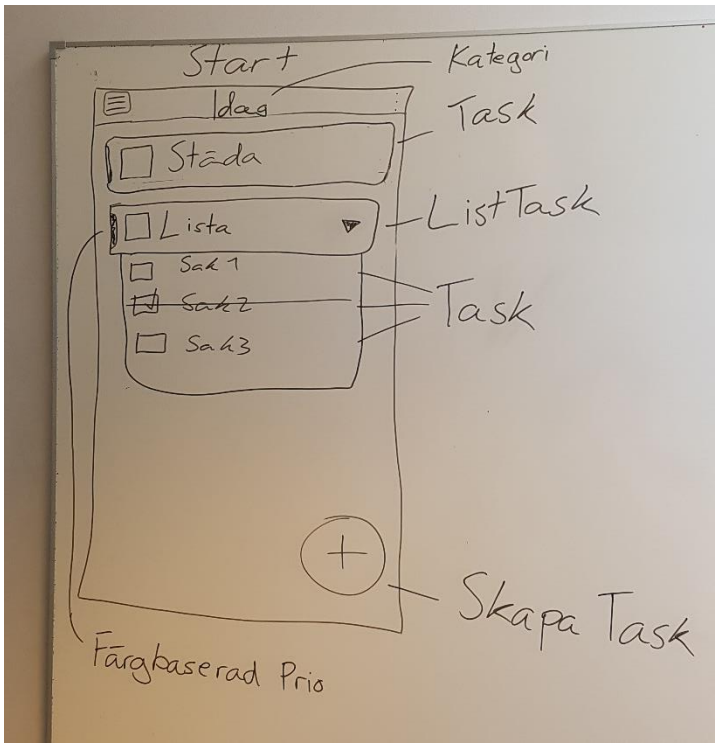
actually shown to the user, and the Controller handles the interaction between the user and the Model.

Object Oriented implementation – A certain form of programming paradigm, where the coding is divided into different objects and classes. This is to break up the different tasks into smaller, more manageable parts and then tackling the problem by creating one “puzzle piece” at a time.

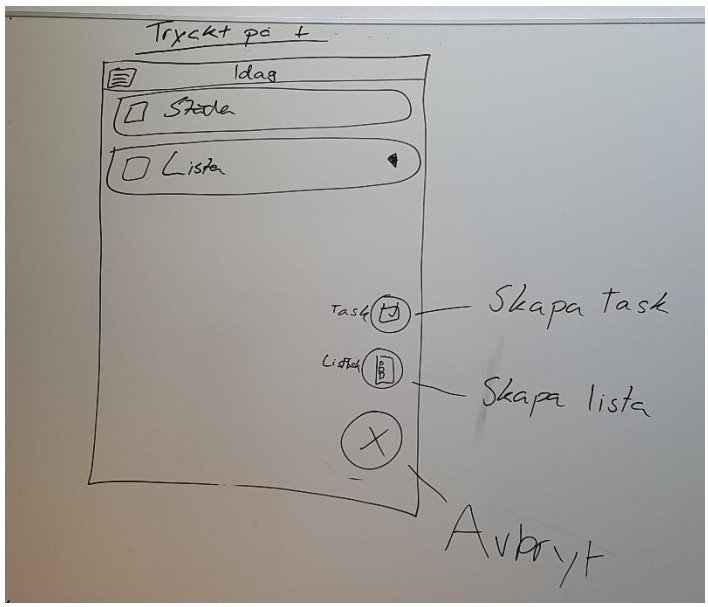
2. REQUIREMENTS

2.1 USER INTERFACE

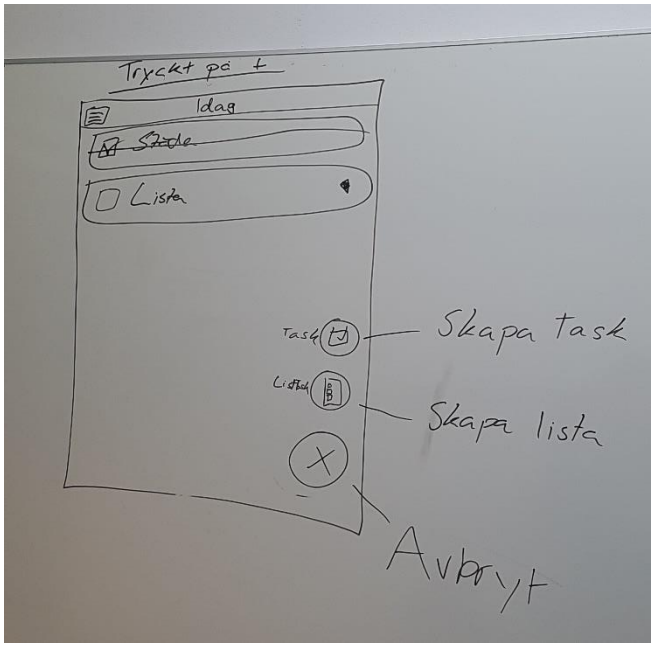
Sketches of the application user interface. The captions describe usage and what it represents.



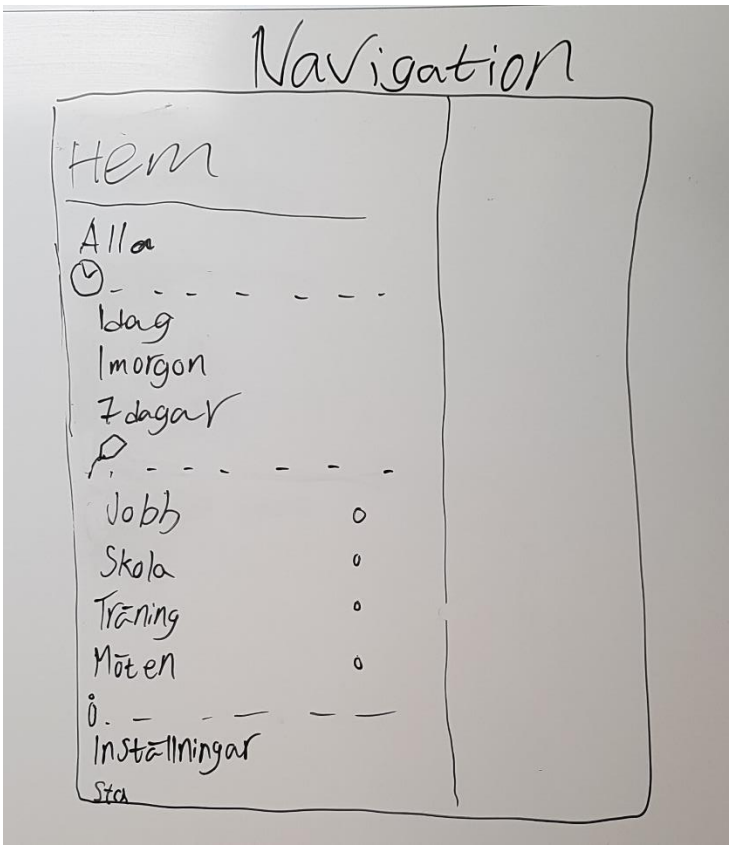
SKETCH 1 - START SCREEN FOR THE APPLICATION, WITH THE DIFFERENT FEATURES MARKED OUT



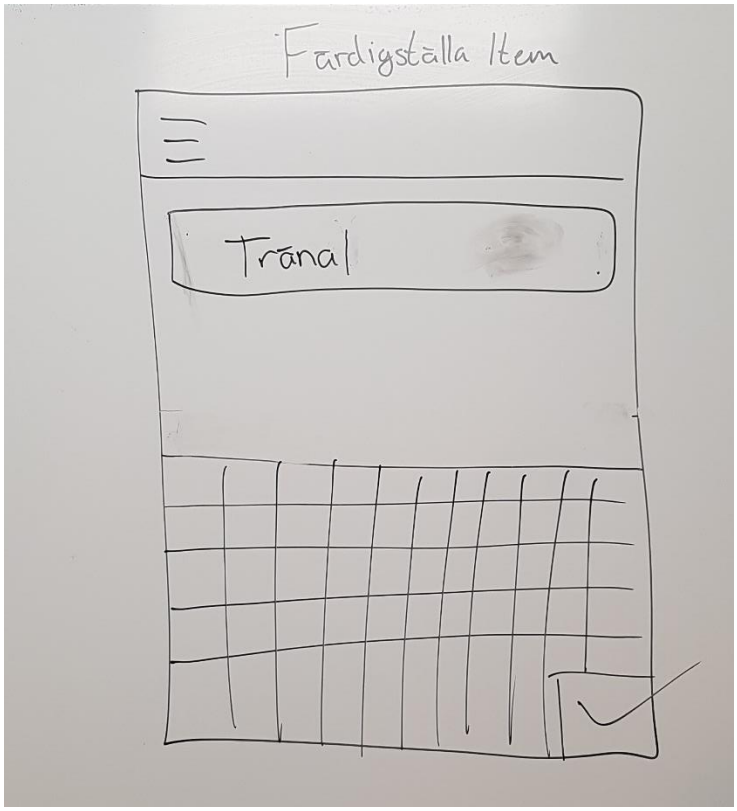
SKETCH 2 - HOME SCREEN VIEW OF THE APPLICATION, WHERE THE USER HAS PUSHED THE "ADD ITEM" BUTTON IN THE LOWER, RIGHT CORNER AND NOW HAS THE CHOICE TO CREATE EITHER A TASK OR A LIST



Sketch 3 - Same Home Screen view, but now the user has checked off a Task, where the system strikes through the task name and adds a checkmark in the checkbox



Sketch 4 - Description for the Navigation Drawer, which is accessed by pushing the upper left button, or sliding from the left side of the screen



SKETCH 5 – VIEW OF THE CREATION OF A TASK, WHERE THE USER TYPES IN A NAME FOR THE TASK AND THEN CLICKS THE CHECKMARK ON THE KEYBOARD TO FINISH THE CREATION

2.2 FUNCTIONAL REQUIREMENTS

The listing below describes the different Use Cases that the user can have in the application, with a prioritizing order. This prioritization is reflecting the different iterations and stages of the application, and therefore what the user will be able to do in each version.

(Each iteration includes the preceding functions from prior priorities.)

FIRST PRIORITY

- **Create a new task in the form of**
  - **General Task**
  - **List Task**
- **Check off a task**
- **Filter the list to a certain category**

SECOND PRIORITY

- **Update a task's configuration**
- **Sort task in another order**
  - **Priority level**
  - **Chronological order**
  - **Alphabetical order**
- **Update/clean up list of done tasks**
- **Delete a task**
- **Create new Label categories through:**
  - **Task creation**
  - **Static category creation**

THIRD PRIORITY

- Change settings
- Check for help
- Check achievements
- Check statistics
- Use Advanced tools when creating Items
    - Set tags for time, priority, category and possibly position

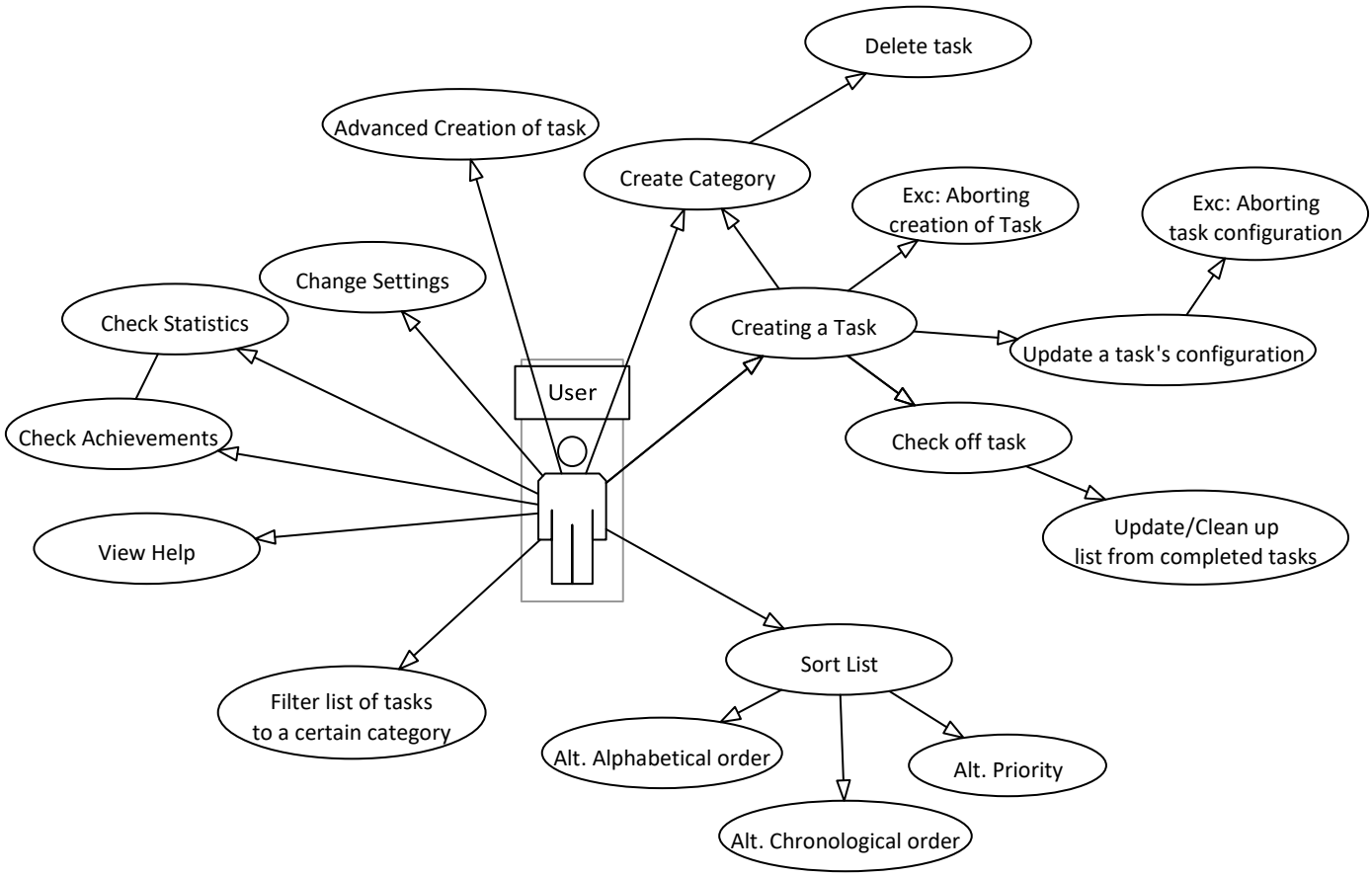
2.3 NON-FUNCTIONAL REQUIREMENTS

The listing below describes the different requirements which the project will fulfill during and after the development process.

- Main Categories
    - Time
    - Labels
- Standard Subcategories
    - ”Today”
    - ”Tomorrow”
    - ”7 days”
    - (”30 days) – Found in Settings
    - ”All”
    - ”Home”
    - ”Work”
    - ”Meetings”
- Support for different Android devices from OS ”Lollipop” and upwards
- Support for both tablets and smartphones
- Implementation through Java
    - Test-driven development
    - MVC-focused implementation model
    - Object Oriented implementation
- Testability
    - Everything that is implemented will be possible to test through JUnit tests.
- Persistence
    - Essential data will be stored after execution for use in upcoming executions.
- Performance
    - Optimized for minimal loading times and instant response
- Usability
    - Simple introduction at first startup of application
    - Focused on simple use, with a clean and modern UI
    - Well optimized for efficient use
    - Implemented functionality for more advanced usage in “Creation mode” for tasks
- Entertainment
    - Make use of gamification through Achievements
    - Create interesting representations for statistics
- Legal
    - All media/data that is used is self-produced

3. USE CASES

This diagram depicts the different Use Cases for the application and also how the different Use Cases coexist.



3.1 USE CASE LISTING

Below you can find the different Use Cases described in a more in-depth manner. Each step describes either the action the user takes, or what the system will respond with to the user's taken action.

Use Cases

First Priority

Creating a Task

State	User	System
1	Press the "Add" button	
2		Shows a list of possible Tasks for the user to create
3	User picks preferred Task	
3.1	Task	
3.2	List Task	
4		Creates an empty Task/List task and Shows the keyboard so the Task can be named
5	Types in a fitting name and confirms the Task creation	
6		Shows the new task in the list, depending on sorting order

Check off Task

State	User	System
1	Taps the empty checkbox	
2		Ticks the checkbox, then strikes through the name

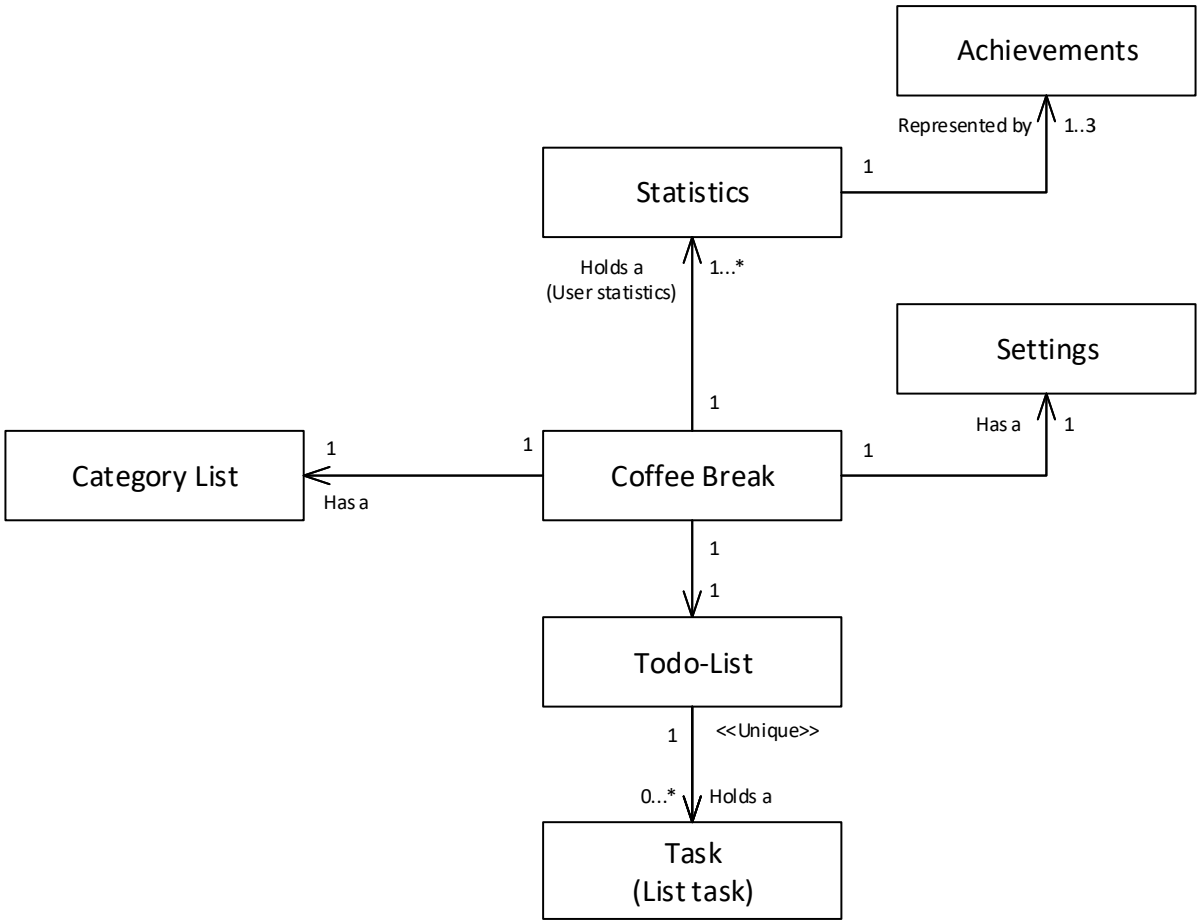
Filter list of tasks to a certain category

State	User	System
1.1	Taps "Navigation" button	
1.2	Swipes from left side of screen to right side to show the Navigation Drawer	
2		Shows the Navigation Drawer with different categories
3.1	Picks a preferred time based category to show the tasks for that timespan	
3.2	Picks a preferred custom category to show the tasks with that label	
4		Slides Navigation Drawer back to left side
5		Sets header to picked category's name, shows all tasks with picked filter

Exception: Aborting creation of a Task

State	User	System
1	Press the "Add" button	
2		Shows a list of possible Tasks for the user to create
3	User picks preferred Task	
3.1	Task	
3.2	List Task	
4		Creates an empty Task/List task and Shows the keyboard so the Task can be named
5.1	Begins to type in a name and then aborts by	
5.1.1	Clicking the checkmark on the keyboard/outside the keyboard	
5.1.2	Clicking the abort button	
5.1.3	Taping "Return"-button	
5.2	Aborts before clicking typing anything	
6.1.1		Shows the new task in the list, depending on sorting order
6.1.2		Removes the task from the list
6.1.3		Removes the task from the list
6.2		Removes the task from the list

4. DOMAIN MODEL



4.1 CLASS RESPONSIBILITIES

Below you will find a more specific description of what each class in the Domain Model above represents. These characteristics are based on properties such as:

- **Multiplicity – Is the object unique or does the occur multiple objects of the same class?**
- **Relationships – Do the classes relate to each other in some way?**
- **Lifecycle – How long does the object live during execution?**
- **Persistence – Will the object persist after the execution?**
- **Mutability – Will the object ever change?**

COFFEE BREAK

- **The main entry point to the Model/database.**

TODO-LIST

- **Unique – Only one object exists**
- **Parent of different tasks**
- **Lifecycle: From start to finish**
- **Persistence: Saves all tasks after execution for next run**
- **Mutable through updates of the list of tasks**

TASK

- **Holds a name**
- **Can be checked off by user**
- **Lifecycle: From creation until checked off or until it is removed**

- Persistence: Gets saved after execution
- Mutable through name editing
- Represents the base for all other tasks
- Can be configured with different settings from the user

(List Task)

- Extension of Task
- Also holds:
  - A list of Tasks in itself
- Mutable by editing list of Base Tasks

Category List

- Unique – Only one object exists
- Holds a complete list of categories in two forms:
  - List of Time Categories
  - List of Label Categories
- Mutable through the addition/removal of categories
- Lifecycle: From start to finish

Statistics

- Unique – Only one object exists
- Manages all the statistics created from application usage
- Mutable through constant statistic updates
- Lifecycle: From start to finish
- Persistence: Persistent through data load and save before and after each lifecycle of the application

Achievements

- Represents different requirements for the Statistics
- Unlocks when a certain criterion in Statistics is fulfilled
- Mutable through two different states: Locked and Unlocked
- Lifecycle: From start to finish
- Persistence: State persists through each execution

5. REFERENCES

The three laws of Test Driven Design - https://blog.jetbrains.com/idea/2016/12/live-webinar-the-three-laws-of-tdd/