REQUIREMENTS AND ANALYSIS DOCUMENT FOR COFFEEBREAK

VERSION: 1.0

DATE: SATURDAY, APRIL 1, 2017

AUTHOR: FELIX NORDÉN

This version overrides all previous versions.

1. INTRODUCTION

Background explaining why this application is needed (besides mandatory in course). What's the problem addressed (use imagination)? What will it do? Who will benefit/use from this application? In what situation will the application be used? Defining the application. General characteristics of application.

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.

1.1 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Create word list to avoid confusion.

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

Base Task - The base of all Tasks types, which is added into List task as their respective items

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 both a category with the List task name and also contains a list of different tasks.

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/ .

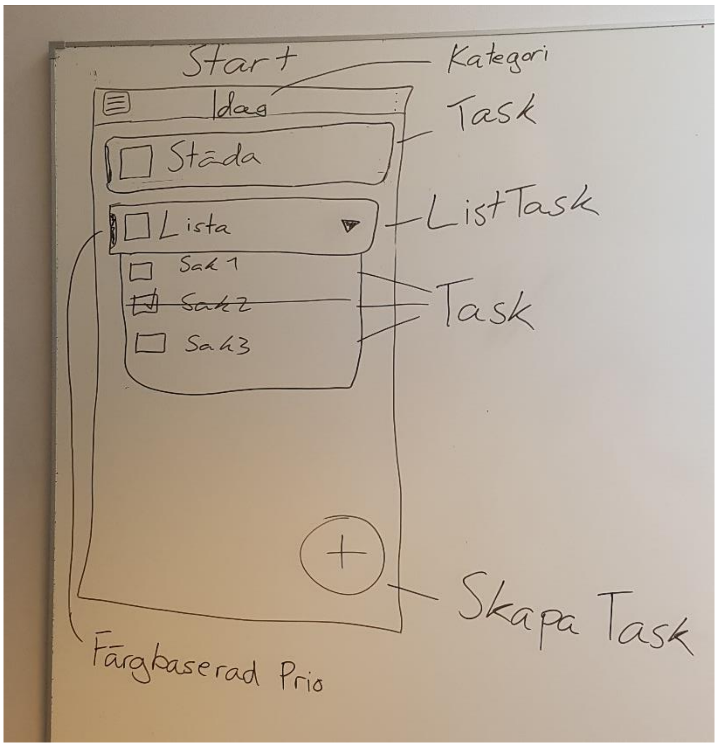
MVC – "Model, View, Controller", a design model used in the most applications/programs today. The Model is the database which handles all of the logic and calculations. The View is the what is actually shown to the user, and the Controller handles the interaction between the user and the Model.

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.

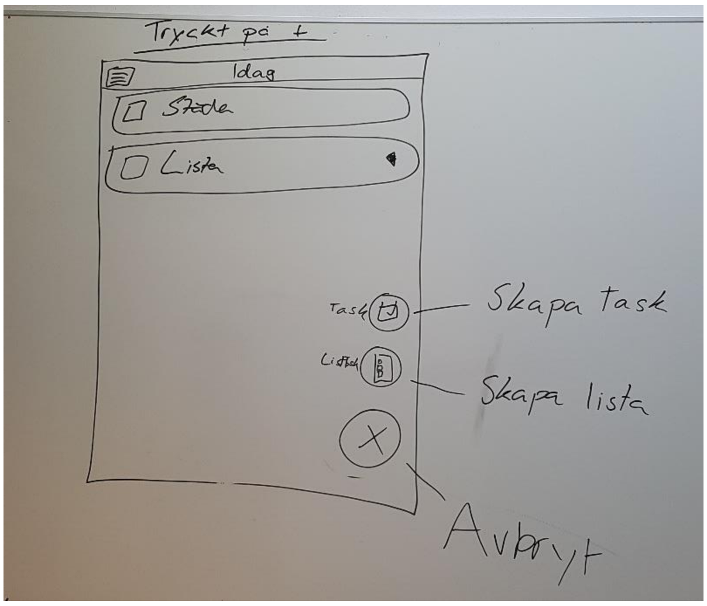
2. REQUIREMENTS

2.1 USER INTERFACE

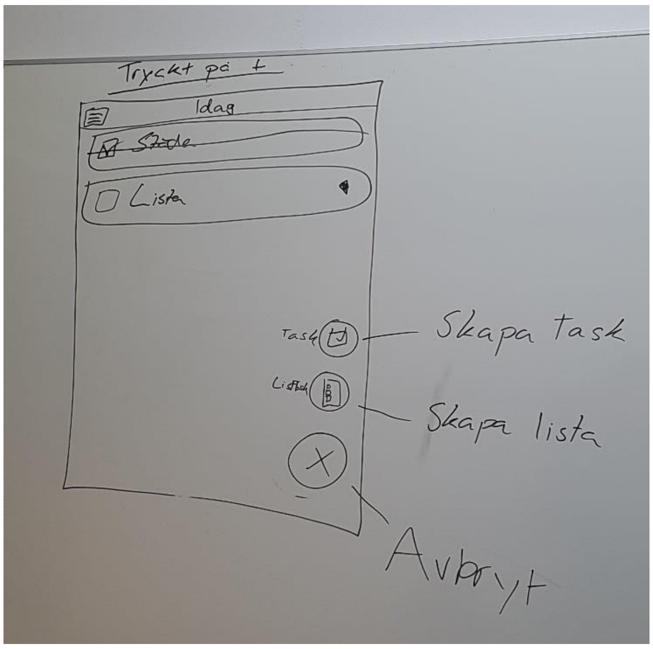
Sketches, drawings and explanations of the application user interface (possible navigation).



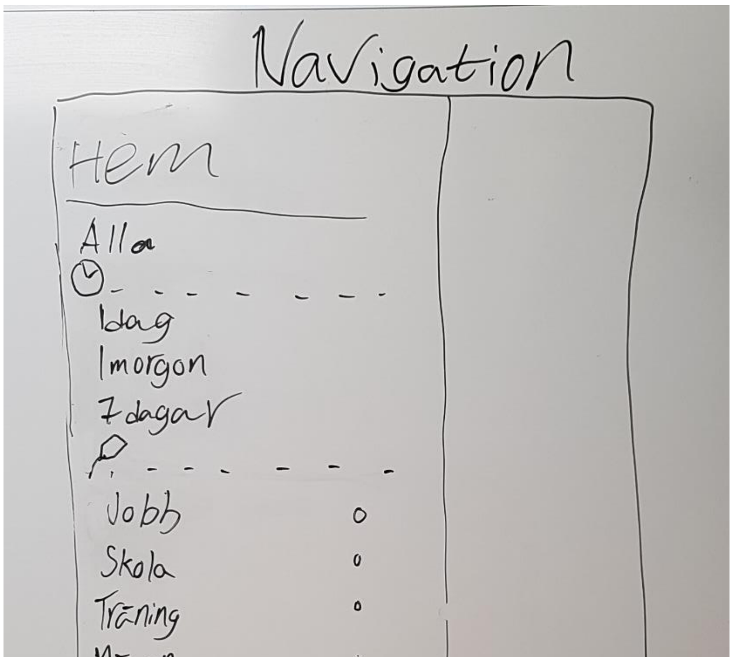
SKETCH 1 - START SCREEN FOR THE APPLICATION, WITH THE DIFFERENT FEATURES MARKED OUT



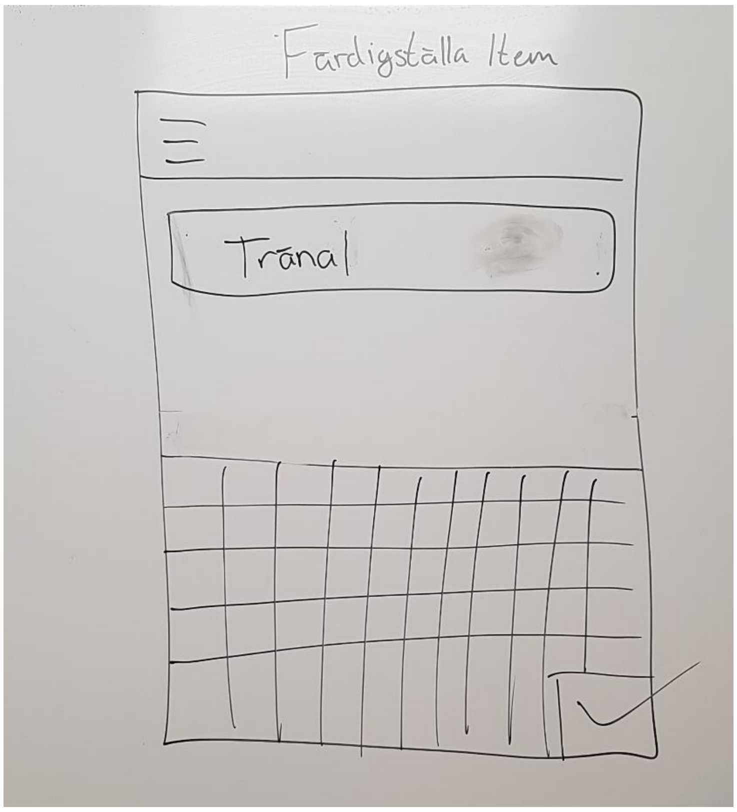
SKETCH 2 - HOME SCREEN VIEW OF THE APPLICATION, WHERE THE USER HAS PUSHED THE "ADD ITEM" BUTTON IN THE LOWER, RIGHT CORNER AND NOW HAS THE CHOICE TO CREATE EITHER A TASK OR A LIST



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

What will the user be able to do ? Write a list of use case names (id's) in the language of the customer. The specific flows for each use case is recorded below. Specify a use cases in priority order.

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 filters/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

Any special considerations besides functionality? Usability, reliability, performance, supportability, legal, implementation, … NOTE: Testability mandatory (must have tests)

- Main Categories
    - Time
    - Labels
- Standard Subcategories
    - ”Today”
    - ”Tomorrow”
    - ”7 days”
    - ”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

See the file ”Use Case Diagram ver. 2” inside the ”UML Diagrams” directory

3.1 USE CASE LISTING

See the Use Cases.xlmx file in this directory.

4. DOMAIN MODEL

See the file ”Domain Model ver. 3” inside the ”UML Diagrams” directory

4.1 CLASS RESPONSIBILITIES

Explanation of responsibilities of classes in diagram

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**

BASE 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.**

TASK

- **Extension of Base Task**
- **Also holds:**
    - **Time/Label categories**

LIST TASK

- **Extension of Task**
- **Also holds:**
    - **A list of Base Tasks**
- **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**

TIME CATEGORY

- **Unique – No identical category names**
- **Holds:**
    - **A name**
    - **Filter for specific time**
- **Exists 3 types: Today, Tomorrow, 7 days forward**
- **Lifecycle: From start to finish**

- Persistence: Hard Coded as of this iteration, so they do not persist after execution
- Immutable – Cannot be edited.

Label Category

- Unique – No identical category names
- Holds:
    - A name
    - Name based filter
- Exists 3 types: Work, Home, Meetings
- Lifecycle: From start to finish
- Persistence: Hard Coded in this iteration, so they do not persist after execution
- Immutable – Cannot be edited, as of this iteration

5. REFERENCES

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