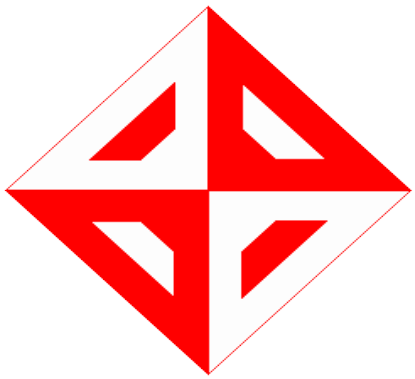


SOFTWARE REQUIREMENTS SPECIFICATION

Version: 1.0
Date: 30.11.2014

Beeminder supported sub applications suite

	Name	ID
Prepared by	**Rufet Eyvazli**	**1645894**
Prepared by	**Esragul Korkmaz**	**1881341**
Prepared by	**Songul Abuzar**	**1678614**
Prepared by	**Alihuseyn Gulmammadov**	**1848282**

Preface

This document contains the system requirements for Android Data Entry Application Suite for Beeminder Project. This document is prepared according to scaled version of the “IEEE Recommended Practice for Software Requirements Specification – IEEE Std 830 –1998” for CENG491.
This Software Requirements Specification provides a complete description of all the functions and specifications of the Android Data Entry Application Suite for Beeminder Project.

The first section of this document includes purpose, scope, references, definitions
and overview of the document.

The second section of this document includes product perspectives, System Interfaces , User Interfaces , Hardware Interfaces , Software Interfaces , Communication Interfaces , Hardware Constraints , Operations , Memory constraints , Application Adaptation Requirements , Product Functions , User characteristics , Constraints , Assumptions and Dependencies , Apportioning of requirements

The third section of this document includes specific requirements, External Interfaces and use cases of the Android Data Entry Application Suite for Beeminder Project.

The fourth section of this document includes data model and description of the project.

The fifth section of this document includes Behavioral Model and Description , Description for software behavior and State Transition Diagrams

The sixth section of this document includes team structure, planning and process model of the project.

The final section of this document includes conclusion of the project.

Change History

Version Number	Date	Brief Description
1.0	30.11.2014	Original

Table of Content

Tables

Figures

INTRODUCTION

This section introduces software requirement specification (SRS) document for Easy Data Entry Applications Suite for Beeminder and it provides complete description of user interaction with system and define all specific information related with system.

1.1 Problem Definition

In our project, we are supposed to implement 6 user -friendly different applications, which makes specific data entries for the Beeminder easier. Beeminder provides data entry manually. However, for a user-friendly usage, it is important to make the submission for data entries easily and enjoyable. Therefore, as team we came together to implement as many as different small applications as a application suite for specific data entries to Beeminder.

1.2 Purpose

Our purpose here is to specify all requirements for this project, considering all components. We will mention about programmer-side, user-side, company-side and hardware-side requirements and information. Our target customers covers all people from any geography, who wants to regulate their lives by using the quantified-self methods.

1.3 Scope

This software will be android platformed phone application for Beeminder users. This application suite will be designed to help the user to

- Keep track of their push-ups.
- Be dominant in their weight -loss control.
- Remind daily task from to-do-list consciously.
- Help to focus on a selected task during a specific time interval.
- Bound Internet usage time and help user to get rid of from wasting time problem.
- Wake-up and take attendance according to fixed schedule.

1.4 Definitions, acronyms, and abbreviations

Definitions, acronyms, and abbreviations	**Definition**
SRS	Software Requirement Specification is a

	document that completely describes all of the functions of proposed system and the constraints under which it must operate.
IEEE	Institute of Electrical and Electronics Engineers.
Android	A mobile device operating system developed by Google Inc
Beeminder	It is quantified self plus commitment application. It is reminders with a sting.
User	Application suit user
Database	Collection of all the information monitored by this system

1.5 References

- IEEE. IEEE Std 829-2008 Standard for Software and System Test Documentation. IEEE Computer Society,2008
- IEEE. IEEE Std 830-1998 Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998
- IEEE. IEEE Std 1016-2009 Standard for Information Technology-System Design-Software Design Descriptions IEEE Computer Society, 2009

1.6 Overview

This document provides detailed description of the Android Data Entry Application Suite for Beeminder project and describes general software and hardware constraints as well as any assumption and dependencies concerning the projects. The majority of this document focuses on the specific requirements list.
This document consists of seven main parts. First section introduces the SRS document and gives brief information about document content. Overall information about the system is provided by the second section. All specific requirements are explained in detail in the third section and all the internal and the external interfaces are addressed in the subsequent of the this section. System data model is described in detail with system database tables in fourth section. Also, table keys and system attributes are briefly described. UML Class Diagram is used for clarifying behavioral system model for next section of document. Section six is used for informing about Android Data Entry Application Suite for Beeminder project's software development method and weekly

scheduled team progress planning. This requirements document concludes with conclusion section of the whole document.

2. Overall Description

This section's aim is to describe and give details about general factors that play main role over whole system and to help to understand the requirements by providing a background.

2.1. Product Perspective

Applications suite will consist of from main three part. The first part will contains Android application which collect all sub applications on one application and create application suite. The second part consist of connection with Beeminder. For easy data entry all sub applications need to make access user account on Beeminder and edit new data entries. The last part will consist of Database connection. For some sub applications inside of application suite needs to keep some user entered data for next usage and increase functionality of application. The project is part of larger system and it is an dependent product. Due to this fact the application consist of user application interaction and in background application database and application Beeminder interactions.

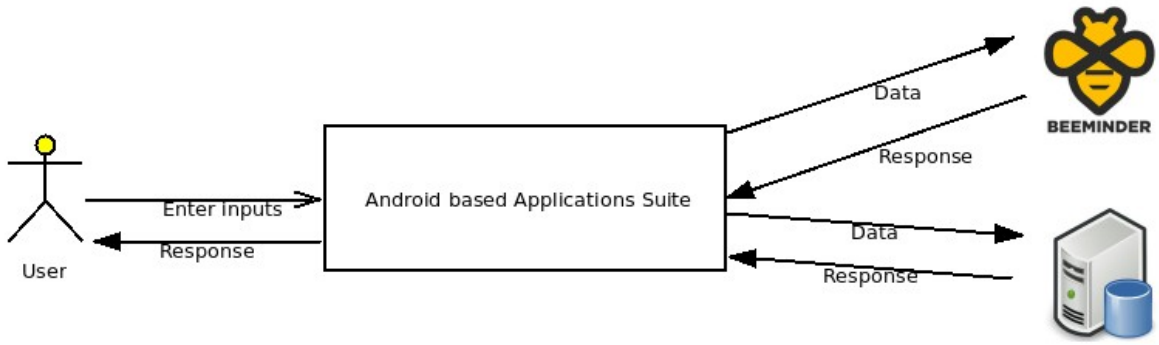


Figure 1. Context Diagram

2.1.1 System Interfaces

As stated above the applications suite is depend which means all facilities can be done over Beeminder connection. TCP/IP Socket Protocols will be used for data communication between Beeminder server and applications suite.

2.1.2 User Interfaces

The project interfaces are developed for Android OS platform and all interfaces will be clear and easy to use. There will be eight main pages and many related sub pages. Initially, user will be faced with the android based applications suite login page.

User can login application suite with using only Beeminder account username and password . Application suite main page will include collection of 6 sub applications redirect button . System will redirect user to selected sub application main page. There will be six sub application main pages in terms of Time Tracker, Push Ups, Weight Loss, Smart Reminder , Habit Control, Schedule Alarm and two or three basic pages including adding new application tasks , updating and deleting related application tasks, showing all tasks and their graphics transformed from Beeminder.

2.1.3. Hardware Interfaces

The only hardware interface requirement is Android OS platformed phone device.

2.1.4. Software Interfaces

This section purpose is to specify the required software products. The required software products and their usage areas over whole project are as belows:

- The applications suite will be based on Android platformed and only be usable on only Android operating system platformed devices.
- The application will be written on Eclipse IDE and Java, Android libraries will be used.
- The interaction with Database will not be seen by user. Appropriate Android libraries will be used for supplying connection with SQLite database.
- The interaction with Beeminder also will not be seen by user and supplied TCP/IP socket protocols from android libraries will be usable.
- To check application usability for different Android based platform Android SDK will be used.

2.1.5. Communication Interfaces

The communication between the applications suite and Beeminder will be provided by TCP/IP protocol. It is supplied inside of android libraries. Database and applications connection will be made available with usage of database connection libraries presented inside of android libraries collection.

2.1.6. Hardware Constraints

The system will be usable for only Android based platform and the only constraint will be usage of Android based devices for applications suite.

2.1.7. Operations

Type of operations is explained in User Interfaces section 2.1.2. Due to that it will not be rewritten again in this section.

2.1.8 Memory constraints

The application will take limited memory usage on phone main memory. However, there will be no limit on user memory usage for entered tasks.

2.1.9. Application Adaptation Requirements

There will not be any adaptation requirements. The applications suite will be user-friendly and usage of these applications will be so easy for user and system is Android based. Anyone who has android platformed device, Beeminder account and Internet can use this applications suite very easily.

2.2 Product Functions

This part will explain higher specification of system regarding to the main functionalities that system will supply for sake of clarification. The below drawn table gives overall brief explanation about sub applications. The detailed explanation will be provided in the next sections.

Login	Provide login to applications suite for existing Beeminder account
Logout	Provide secure logout from applications suite
Time Tracker	Keep time for different work facilities and make to focus on current work for selected time interval.
Push Ups	Control push-up with set and provide a feedback with graph taken from Beeminder

Weight Loss Control	Make easier data entry with pictures of scales and inform changes for weight
Smart Reminder	Consciously remind tasks from Beeminder or from existing database and get your reply and provide a new data entry.
Habit Control	Try to make restriction on Internet usage with working background and provide spent time as a new data entry.
Schedule Alarm	Provide voice recognized alarm system and keep track of attendance for added schedule.

Table 1. Brief Functional Requirements

2.3 User characteristics

The user is expected to be a mid-level smartphone user and be able to use a Android based platform.

2.4 Constraints

- The user is expected to have a Beeminder account.
- Android framework and Java knowledge is mandatory.
- Developer of mobile platforms is needed to care about memory issues because of limited memory, that the applications suite will cover.
- The security of the login authentication and store transactions should be provided by Beeminder security system. The reliability of data communication should be provided by TCP/IP.
- Changing data entries transformed by Beeminder tasks should be updated by Beeminder automatically.

2.5 Assumptions and Dependencies

Since this is an Android application suite project and it contains many functionalities and opportunities, some other functionalities may be popular and then these features can be added to the project in the future for more attractiveness and more usability and those will affect server database or user interfaces.

2.6 Apportioning of requirements

There will be not any delayed requirements until future version of the system.

3. Specific Requirements

All functional and nonfunctional requirements of the system are described in detailed in this section. Any designer will be capable of design all system thanks to these detailed requirements parts.

3.1 External Interfaces

This section supply detailed information about requirements that mentioned in section 2. Sketched mock-ups describe user interface with required controls.

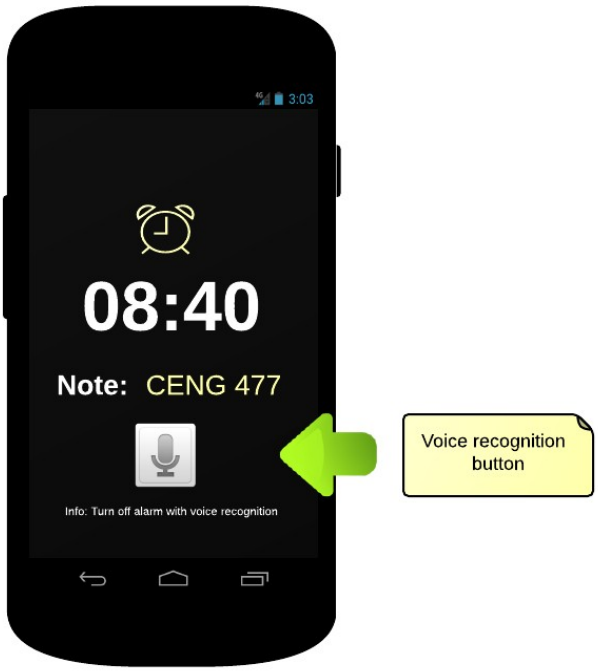


Figure 2. Schedule Alarm mock-up

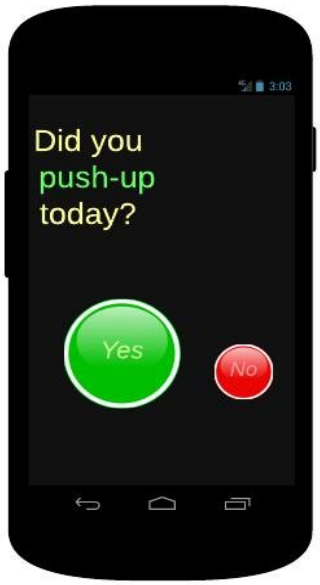


Figure 3. Smart Reminder mock-up

Figure 4. Start menu of applications suite mock-up

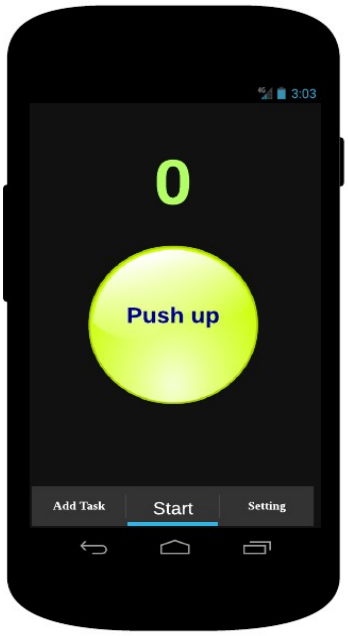


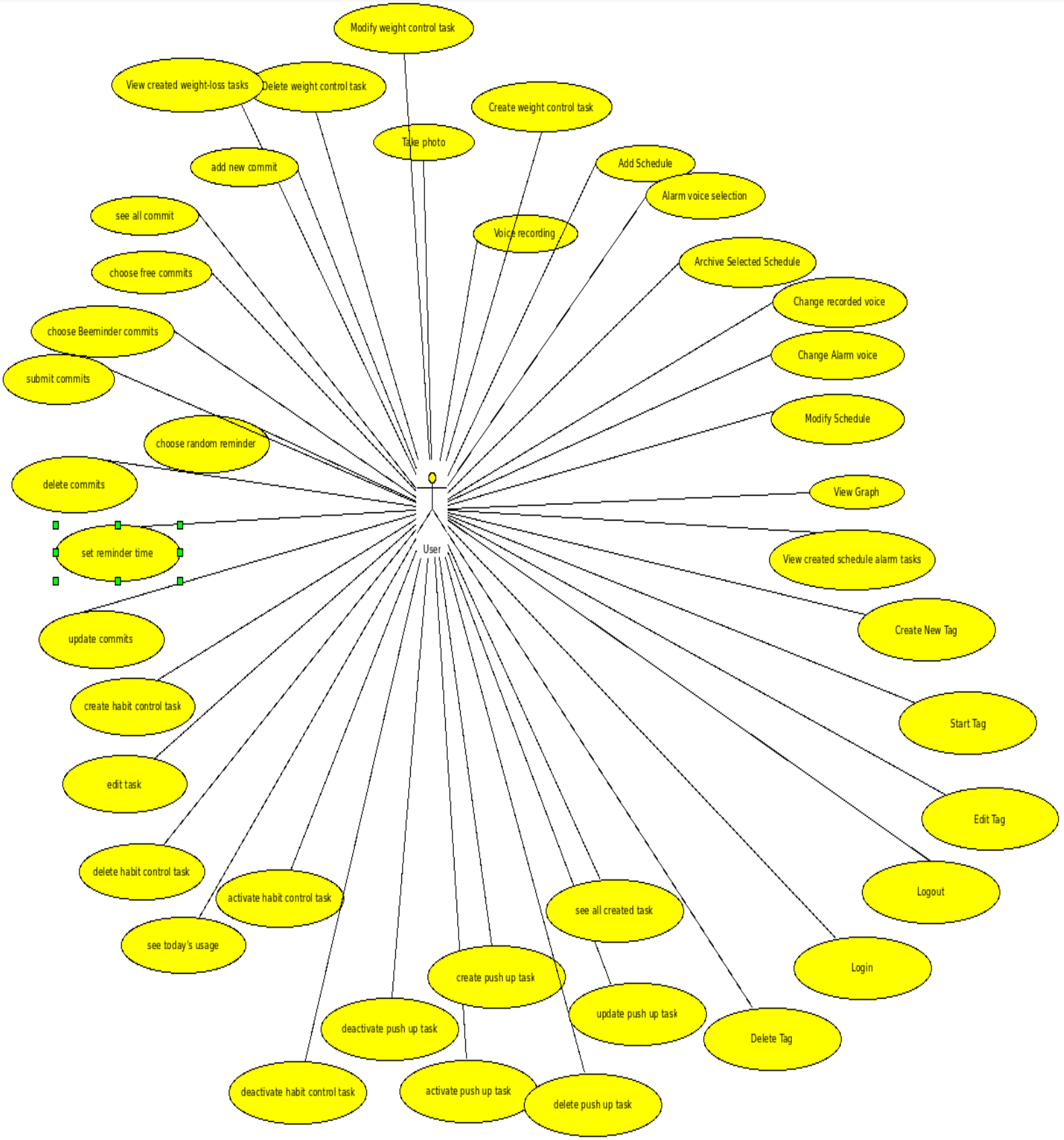
Figure 5. Push-ups mock-up

Figure 6. Time Tracker mock-up

Figure 7. Habit Control mock-up

3.2 Functional Requirements

Figure 8. Use case diagram



3.2.1. Use Case: Voice Recording

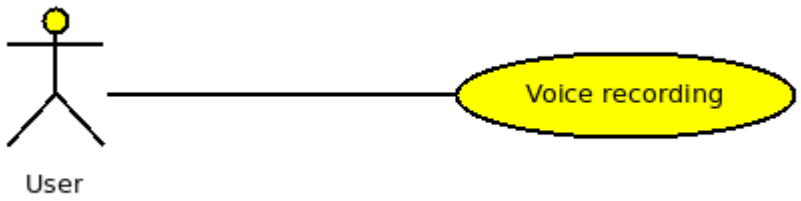


Figure 9. Use case diagram for Voice recording

Brief Description:

User can record his voice for schedule alarm application with this use case while creating new schedule item. The recorded voice will help to disable alarm with voice recognition.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Schedule Alarm button and the system will automatically switch user to selected application layout.
3. The system will provide a button Add new entry. A click on it will redirect user to a new menu included Voice recording case.
4. The system will provide voice recording features for user and will keep recorded voice in the memory of application.

3.2.2. Use Case: Add Schedule

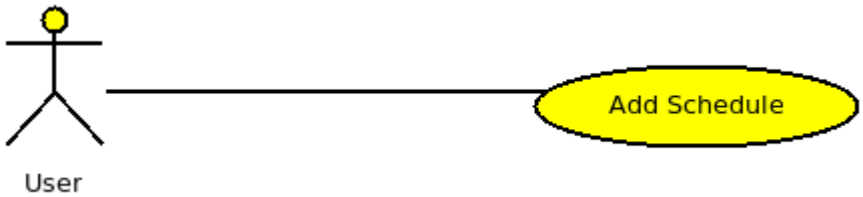


Figure 10. Use case diagram for Add Schedule

Brief Description:

User can add new schedule item for Schedule Alarm application with this use case.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Schedule Alarm button and the system will automatically switch user to selected application layout.
3. The system will provide a button Add new entry. A click on it will redirect user to a new menu included Add Schedule case.
4. The system will provide add Schedule features for user and will keep added schedule in the memory of application.

3.2.3. Use Case: Alarm Voice Selection

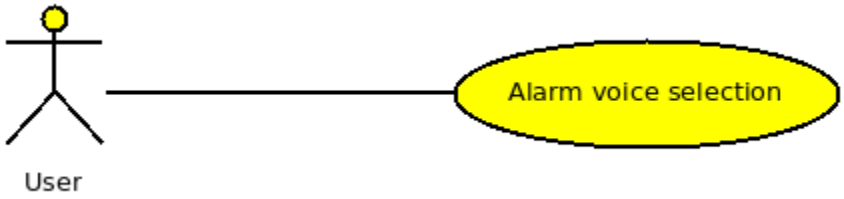


Figure 11. Use case diagram for Alarm voice selection

Brief Description:

User can select alarm voice for added new schedule item with this use case and selected voice will be used as alarm sound.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Schedule Alarm button and the system will automatically switch user to selected application layout.
3. The system will provide a button Add new entry. A click on it will redirect user to a new menu included Alarm voice selection case.

4. The system will provide alarm voice selection features for user and will keep selected entry in the memory of application.

3.2.4. Use Case: Archive Selected Schedule

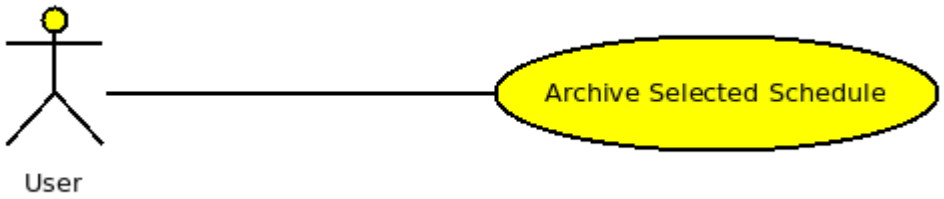


Figure 12. Use case diagram for Archive Selected Schedule

Brief Description:

User can archive selected schedule from settings with this use case. Beeminder doesn't supply pure delete option for user but archive option is available when you deleted any entry.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Schedule Alarm button and the system will automatically switch user to selected application layout.
3. The system will provide a button Setting. A click on it will redirect user to a new menu included Archive Selected Schedule case.
4. The system will provide archive selected schedule features for user and will delete the entry from alarm list and keep it in Beeminder memory as archive.

3.2.5. Use Case: Change Alarm Voice

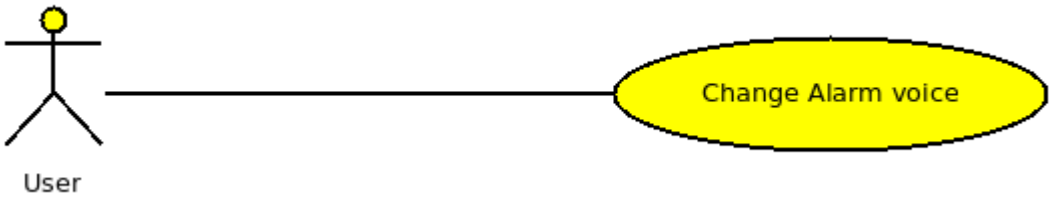


Figure 13. Use case diagram for Change Alarm voice

Brief Description:

User can change alarm sound from settings with this use case.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Schedule Alarm button and the system will automatically switch user to selected application layout.
3. The system will provide a button Setting. A click on it will redirect user to a new menu included Change Alarm voice case.
4. The system will provide change alarm voice features for user and selected voice will be updated for selected entry.

3.2.6. Use Case: Change Recorded Voice

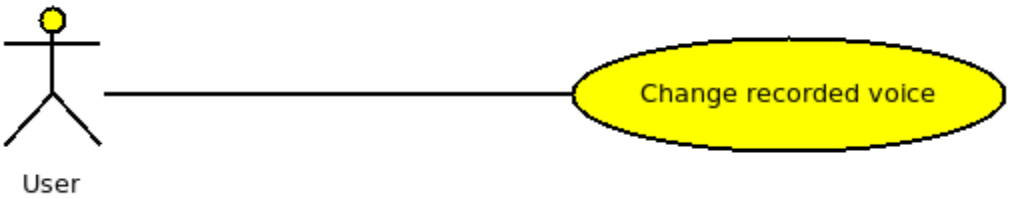


Figure 14. Use case diagram for Change recorded voice

Brief Description:

User can change recorded voice from settings with this use case. The next voice recognition the new changed sound will be used for comparison.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Schedule Alarm button and the system will automatically switch user to selected application layout.
3. The system will provide a button Setting. A click on it will redirect user to a new menu included Change Recorded Voice case.
4. The system will provide change recorded voice features for user and recorded voice will be updated for selected entry.

3.2.7. Use Case: Modify Schedule

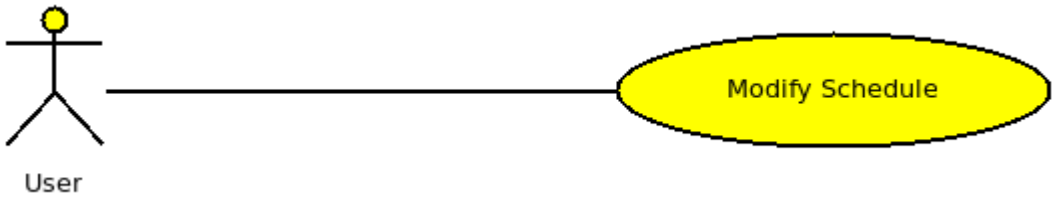


Figure 15. Use case diagram for Modify Schedule

Brief Description:

User can modify schedule from settings with this use case. It will be supplied for user to change goal name or do other changes over selected task.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Schedule Alarm button and the system will automatically switch user to selected application layout.

3. The system will provide a button Setting. A click on it will redirect user to a new menu included Modify Schedule case.
4. The system will provide modify schedule features for user and modified schedule will be updated for selected entry.

3.2.8. Use Case: View Graph

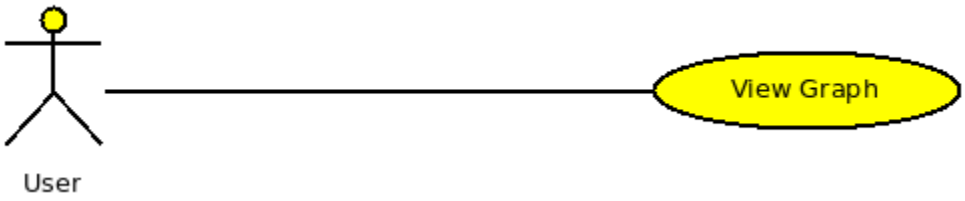


Figure 16. Use case diagram for View Graph

Brief Description:

User can use view graph use case from the main menu of all applications listed in application suite. This case will provide user to observe graphs for selected tasks.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects any application from applications suite and the system will automatically switch user to selected application layout.
3. The system will provide a view graph button. From this button selection , user can reach graph and can observe weekly, monthly, and yearly track on selected entry.

3.2.9. Use Case: Create Weight Control task

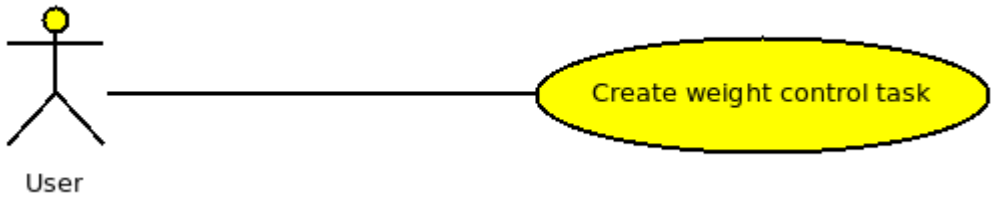


Figure 17. Use case diagram for Create weight-loss control task

Brief Description:

User can create new weight - loss control task with this use case.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Weight-loss control button and the system will automatically switch user to selected application layout.
3. The system will provide Create Weight-loss control task use case. Create a weight-loss control task will be supplied with the help of Beeminder connection for a user.

3.2.10. Use Case: Take Photo

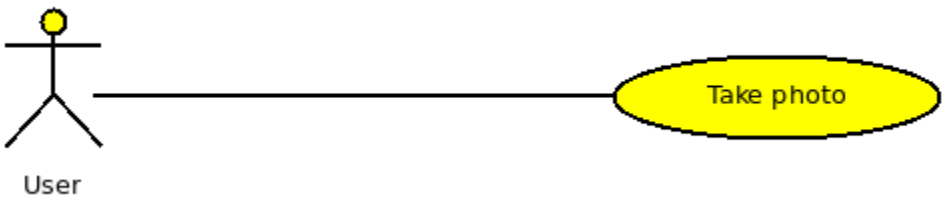


Figure 18. Use case diagram for Take photo

Brief Description:

User can add new entry with Take photo button. The photo will be taken from scales and in background it will converted to a number and added Beeminder account as a new data for weight-loss goal.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Weight-loss control button and the system will automatically switch user to selected application layout.
3. With the help of Take Photo button user takes photo of his weight from scales.
4. In background taken photo will be converted to a number with the help of optical character recognition libraries.
5. Generated number will be added Beeminder account as a new entry.

3.2.11. Use Case: Modify weight control task

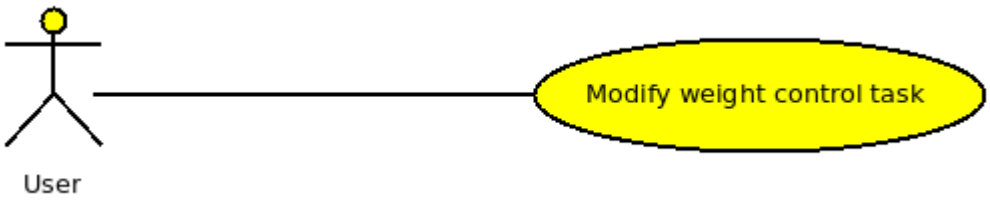


Figure 19. Use case diagram for Modify weight-loss control task

Brief Description:

User can modify selected weight-loss control task . The changes due to goals can be edited in this part.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Weight-loss control button and the system will automatically switch user to selected application layout.
3. From setting menu user can achieve modify weight-loss control task.

4. User can change or reedit to his goals for selected weight-loss control task.

3.2.12. Use Case: Delete weight control task

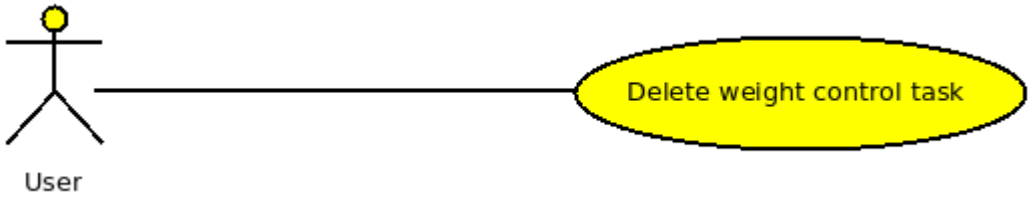


Figure 20. Use case diagram for Delete weight-loss control task

Brief Description:

User can delete weight-loss control task. In Beeminder the deletion operation will archive it and new goal entries for this task will be stopped.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Weight-loss control button and the system will automatically switch user to selected application layout.
3. From setting menu user can achieve delete weight-loss control task.
4. The task will be cleaned from application and kept as archive in users' Beeminder account.

3.2.13 Use Case: View created schedule alarm tasks

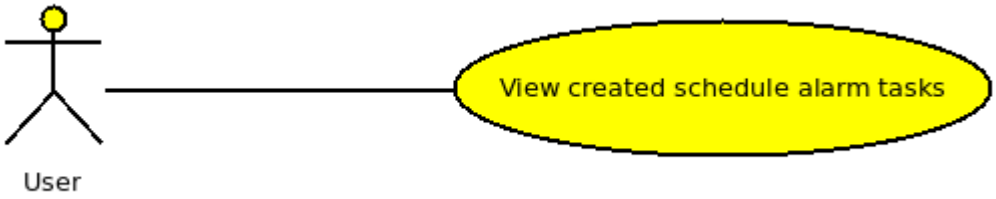


Figure 21. Use case diagram for View created schedule alarm tasks

Brief Description:

The list of all created tasks or goals will be shown as a list with brief information related with tasks while opening Schedule Alarm application. The View created schedule alarm tasks will be achieved by click on the selected task form list. The wide information will be supplied with this selection.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Schedule Alarm button and the system will automatically switch user to selected application layout.
3. In main layout the list of tasks will be shown
4. By clicking on selected task will redirect user to new layout and provide wide information about task for user.
5. View created schedule alarm task is achieved by user.

3.2.14 Use Case: View created weight-loss control tasks

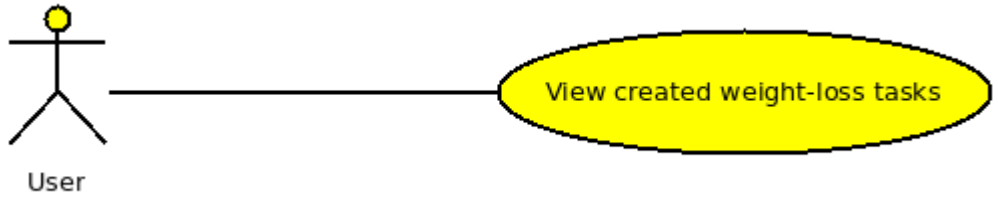


Figure 22. Use case diagram for View created weight-loss control tasks

Brief Description:

The list of all created tasks or goals will be shown as a list with brief information related with tasks while opening Weight-loss Control application. The View created weight-loss control tasks will be achieved by click on the selected task form list. The wide information will be supplied with this selection.

Step-by-Step Description:

1. Firstly, user successfully logins to application suite.
2. User selects Weight-loss control button and the system will automatically switch user to selected application layout.
3. In main layout the list of tasks will be shown
4. By clicking on selected task will redirect user to new layout and provide wide information about task for user.
5. View created weight-loss control task is achieved by user.

3.2.15. Use Case: Create habit control task

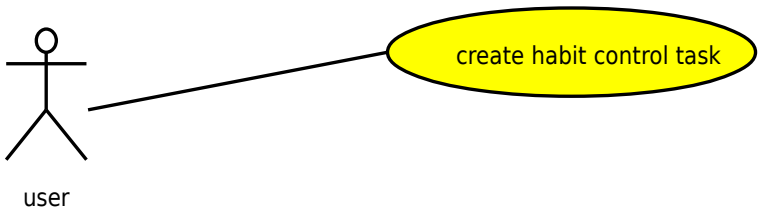


Figure 23. Use case diagram for Create habit control task

Brief Description :

A user can create a new task about daily Internet usage time, and set the time interval to apply this task. He/she can also give this task a specific name. If the user wants to be informed when he/she reaches the limit time, he/she can choose these alternatives here.

Step by Step description :

1. At the main page, user can select the “create a new task”.
2. Here, the user will see a screen to determine the task's time interval, daily time limit and its name. Here, also he/she can decide whether there will be a notification and/or alarm, when he/she reaches the limit time or not. By clicking create button, user can create this task.
3. There will appear success/unsuccessful message in the screen.

3.2.16 Use Case: Update habit control task

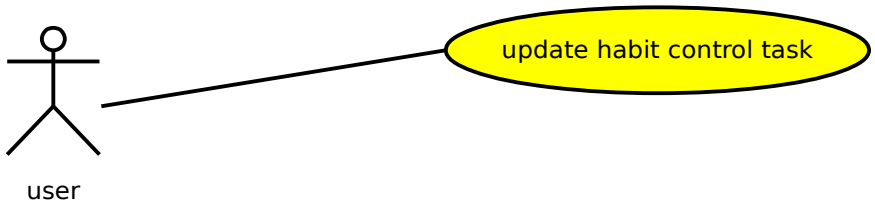


Figure 24. Use case diagram for Update habit control task

Brief Description :

A user can update an existing task about daily Internet usage time. That is, he/she can change the time interval to apply this task. He/she can also change this task's name. If the user wants to be informed when he/she reaches the limit time, he/she can choose to change these alternatives here.

Step by Step description :

1. At the main page, user can select the “update task”.
2. There will appear a screen which shows all created task names .
3. By selecting one of them, user can change his/her goal time interval, daily time limit, task's name, by clicking update button. In addition he/she can change his/her decision about being informed when he/she reaches the daily time limit, by choosing one or more of the three choices, named “do not inform”, “alarm”, and “notification”.
4. There will appear success/unsuccessful message in the screen.

3.2.17 Use Case: Delete habit control task

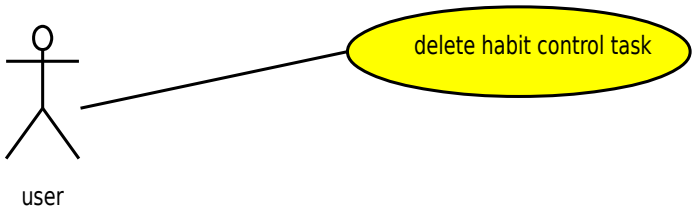


Figure 25. Use case diagram for Delete habit control task

Brief Description :

A user can delete an existing task here.

Step by Step description :

1. User can choose “delete task ” button at the main page.
2. There will appear a screen which shows all created task names .
3. By choosing one of the names, user can delete whichever he/she wants.

3.2.18 Use Case: See today's usage

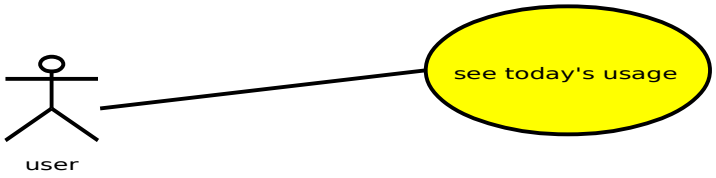


Figure 26. Use case diagram for see today's usage

Brief Description :

A user can see the time of his/her total Internet usage, spent time in each website, connection and disconnection times through an attractive user interface.

Step by Step description :

1. At the main page, user can choose “Daily Data” button to see the data of today.

2. There will appear all tasks in the screen, so that user can choose one of them to see the daily information of this task.

3.2.19. Use Case: See Beeminder graphics

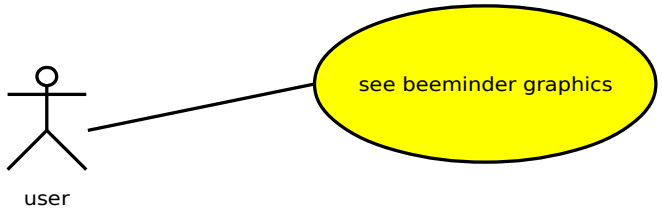


Figure 27. Use case diagram for See Beeminder graphics

Brief Description :

A user can see a task's all Beeminder graphs on the screen, which includes all times from beginning of this task to today.

Step by Step description :

1. User will select the “See Beeminder Graph” button at the main page.

2. The user will be directed to a screen, which shows all tasks created before.

3. He/she can choose whichever he/she wants to see the task's Beeminder graphs in the screen.

3.2.20. Use Case: Activate habit control task

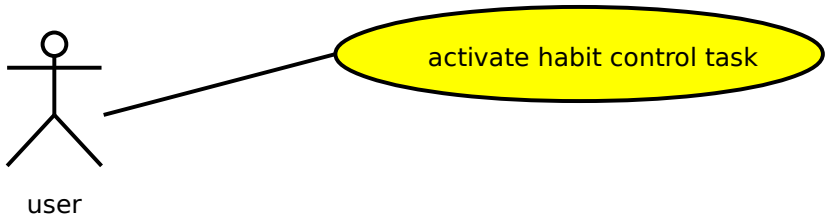


Figure 28. Use case diagram for Activate habit control task

Brief Description :

A user can activate a task, whenever he/she wants.

Step by Step description :

1. When the user selects “activate” button at the main page, there appears a new screen full of all task names.

2. Here, the user can select activate button to activate any task he/she wants, which probes all other tasks to be deactivated temporarily.

3. There will appear success/unsuccessful message in the screen.

3.2.21. Use Case: Deactivate habit control task

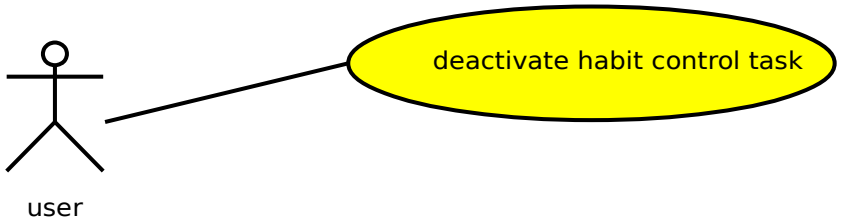


Figure 29. Use case diagram for deactivate habit control task

Brief Description :

A user can take a break or deactivate a task for a short time, whenever he/she wants.

Step by Step description :

1. When user clicks the “Deactivate” button at the main page, he/she will be directed to a page with all task names (activated task is specified if there exists).

2. By clicking the task name, user can deactivate any task he/she wants. There will be two buttons here. If he/she clicks the deactivate button, it means that today's data will be sent to Beeminder. For example, if someone else is using the Internet at the current device, the user can deactivate it temporarily.

3. Here, instead of the “deactivate” button, if user clicks the “take a break” button, this means, the data will not be sent to Beeminder for a time because the user will take a break, for example for two weeks duration.

4. There will appear success/unsuccessful message in the screen.

3.2.22. Use case: Create push-up task

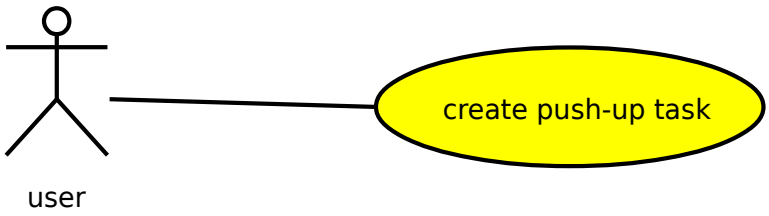


Figure 30. Use case diagram for create push-up task

Brief Description :

A user can create a new task about push up amount, and set the time interval to apply this task. He/she can also give this task a specific name. If the user wants to be informed when he/she does not satisfy daily requirements, he/she can choose these alternatives here.

Step by Step description :

1. At the main page, user can select the “create a new task”.

2. Here, the user will see a screen to determine the task's time interval, and the task's name. Here, also he/she can decide whether there will be a notification and/or alarm, when he/she does not satisfy daily requirements. By clicking create button, user can create this task.
3. There will appear success/unsuccessful message in the screen.

3.2.23. Use case: Update push-up task

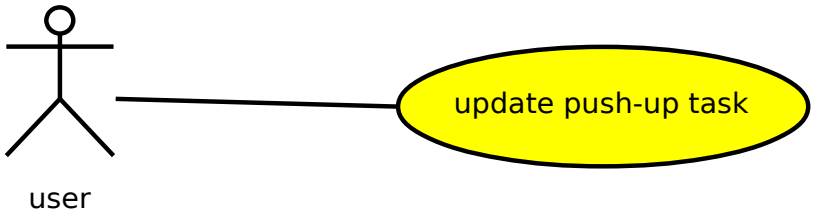


Figure 31. Use case diagram for update push-up task

Brief Description :

A user can update an existing task here. That is, he/she can change the time interval to apply this task. He/she can also change this task's name. If the user wants to be informed when he/she does not satisfy daily requirements, he/she can choose to change these alternatives here.

Step by Step description :

1. At the main page, user can select the “update task”.
2. There will appear a screen which shows all created task names .
3. By selecting one of them, user can change his/her goal time interval, task's name,and make all other possible modifications here. In addition he/she can change his/her decision about being informed when he/she does not satisfy daily requirements, by choosing one or more of the three choices, named “do not inform”, “alarm”, and “notification”.
4. There will appear success/unsuccessful message in the screen.

3.2.24. Use case: Delete push-up task

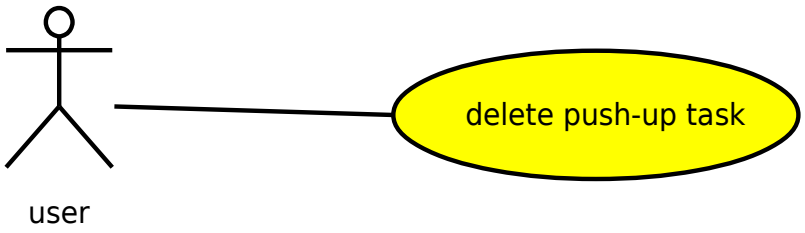


Figure 32. Use case diagram for delete push-up task

Brief Description :

A user can delete an existing task here.

Step by Step description :

1. User can choose “delete task ” button at the main page.
2. There will appear a screen which shows all created task names .
3. By choosing one of the names, user can delete, whichever he/she wants.

3.2.25. Use case : Activate push-up task

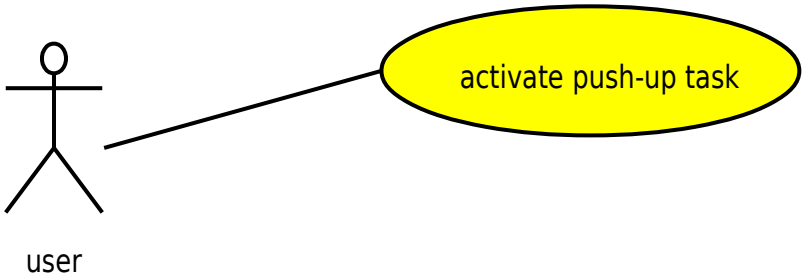


Figure 33. Use case diagram for activate push-up task

Brief Description :

A user can activate a task, whenever he/she wants.

Step by Step description :

1. When the user selects “activate” button at the main page, there appears a new screen full of all task names.

2. Here, the user can select activate button to activate any task he/she wants, which probes all other tasks to be deactivated temporarily.

3. There will appear success/unsuccessful message in the screen.

3.2.26. Use case: Deactivate push-up task

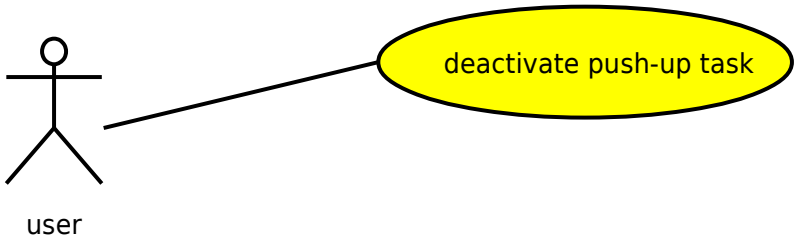


Figure 34. Use case diagram for deactivate push-up task

Brief Description :

A user can take a break or deactivate a task for a short time, whenever he/she wants.

Step by Step description :

1. When user clicks the “Deactivate” button at the main page, he/she will be directed to a page with all task names (activated task is specified if there exists).

2. Here, if user clicks the “take a break” button by choosing an active task, this means, the data will not be sent to Beeminder for a time because the user will take a break, for example for two weeks duration.

3. There will appear success/unsuccessful message in the screen.

3.2.27. Use case: See all created tasks

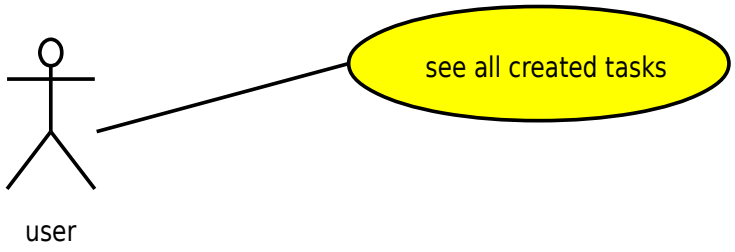


Figure 35. Use case diagram for see all created task

Brief Description :

A user can see all archived tasks here.

Step by Step description :

1. When user clicks the “See archive” button at the main page, he/she will be directed to a page with all previous task names.

2. Here, if user clicks the “detailed info” button by choosing a task, this means, he/she can see all data about it in a new page.

3.2.28. Use case: See all commits

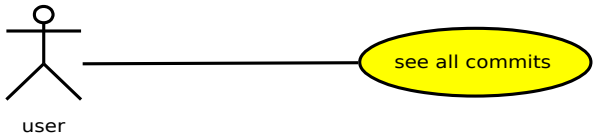


Figure 36. Use case diagram See all commits

Brief Description:

This Use Case provides user to see all commits in list format with brief description of commits in terms of commit name, commit submission date and commit reminder time. It is enough that the user successfully logins to the system and selects the Smart Reminder Application button.

Step-by-Step Description:

1. User starts application and successfully login to the system.
2. User clicks “Smart Reminder” button on the system main page and system redirects the user to Smart Reminder application main page.
3. In this Smart Reminder main page, user can see all commits and their brief descriptions in terms of commit name, commit submission date and commit reminder date. User can achieve all commit informations by clicking one of commits list element.

3.2.29. Use Case: Add new Commits

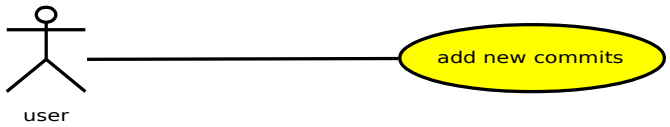


Figure 37. Use case diagram for add new commits

Brief Description:
User can add new commits with this use case. To add new commits , user must login to system and and enter the Smart Reminder main page.

Step-by-Step Description:

1.User logins to the system and selects “Smart Reminder” button.
2. User clicks “New Commit” button on the top of Smart Reminder main page and system redirects user to add new commits page.
3. User initially must choose commit type with clicking “Beeminder commits” or “Free commits” button.
4. If user selects “Beeminder commits” type, user enters Beeminder username and password and shows Beeminder task.
5. User selects any of Beeminder task and clicks “Commit” button near the tasks. User will be add selected Beeminder task as a reminder commit.
6. If user selects “Free commits” type, system redirected user o add new free commits page.
7. Int his page user can add new free commit by entering commit name, commit reminder time and then clicks “Commit” button .

3.2.30. Use Case: Choose Beeminder commits

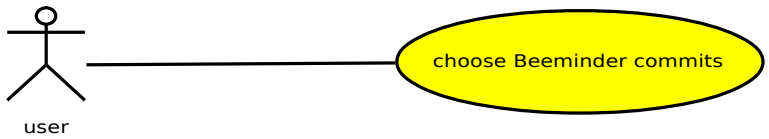


Figure 38. Use case diagram for choose Beeminder

Brief Description:

This use case diagram provides user to select the Beeminder tasks as a reminder commit. Smart Reminder application is integrated with Beeminder quantified self application. With using choose Beeminder commits functionality, user can select all Beeminder tasks as a Smart Reminder commit. Also, it provides automatic data submission to Beeminder tasks.

Step-by-Step Description:

1.User logins to the system and selects “Smart Reminder” button.
2. User clicks “New Commit” button on the top of Smart Reminder main page and system redirects user to add new commits page.
3. User clicks “Beeminder Commits” button on this page and s/he can select tasks on Beeminder to Smart Reminder commit.

3.2.31. Use Case: Choose free commits

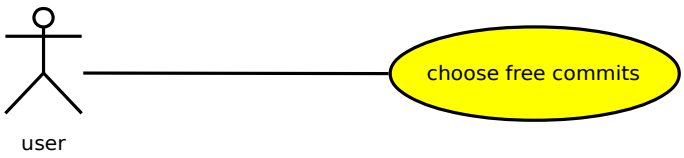


Figure 39. Use case diagram for Choose Free Commit

Brief Description:

When user wants to create new free commits, s/he uses this functionality of the application. To use this use case, user must login to system, select Smart Reminder application and click “New Commit” button.

Step-by-Step Description:

1.User logins to the system and selects “Smart Reminder” button.
2. User clicks “New Commit” button on the top of Smart Reminder main page and system redirects user to add new commits page.
3. User clicks “Free Commits” button on this page and then s/he can add new free commits.

3.2.32. Use Case: Set reminder time

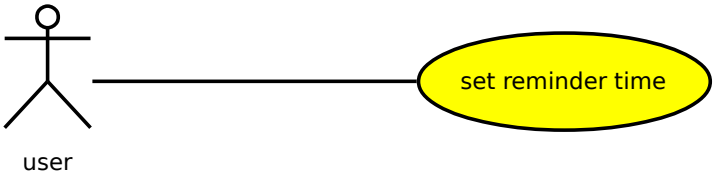


Figure 40. Use case diagram for Set Reminder Time

Brief Description:

This use case provides user to set reminder time manually. User sets reminder time while adding new free commits or updating free commits.

Step-by-Step Description:

1.User login to the system and selects “Smart Reminder” button.
2. User clicks “New Commit” button on the top of Smart Reminder main page and system redirects user to add new commits page.
3. After entering commit name, user selects reminder time in reminder time field.
4. When user enters the Smart Reminder page, s/he selects commit in the commit list.
5. User clicks “Modify Commit” button on the detailed commit page.
6. User changes reminder time in the reminder time field and clicks “Save” button and reminder time will be selected manually.

3.2.33. Use Case: Choose Random Reminder

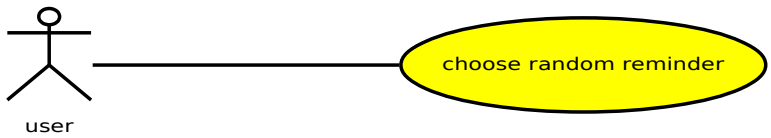


Figure 41. Use case diagram for Choose Random Reminder

Brief Description:

This use case provides user to set reminder time as randomly. User can set free commits reminder type as random when updating or adding free commits. All Beeminder commits are reminded randomly. Beeminder tasks have reminder date for each task. According to these reminder date, Smart Reminder reminds randomly this task to user every day.

Step-by-Step Description:

1.User logins to the system and selects “Smart Reminder” button.
2. User clicks “New Commit” button on the top of Smart Reminder main page and system redirects user to add new commits page.
3. After entering commit name, user selects clicks “Random Reminder” button and then clicks “Save” button.
4. When user enters the Smart Reminder page, s/he selects commit in the commit list.
5. User clicks “Modify Commit” button on the detailed commit page.
6. User clicks “Random Reminder” button and clicks “Save” button and reminder time will be selected as random.

3.2.34. Use Case: Update Commits

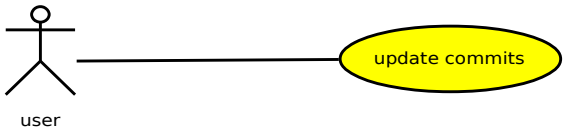


Figure 42. Use case diagram for Update Commits

Brief Description:

User can update any selected commits.

Step-by-Step Description:

1. User logins to the system and selects “Smart Reminder” button.
2. User clicks “New Commit” button on the top of Smart Reminder main page and system redirects user to add new commits page.
3. User clicks linked commit area in the commits list.
4. User clicks “Modify Commit” button on the detailed commit page.
5. There are “Commit name” and “Commit reminder time” areas in this field. User can change these properties and clicks “Save” button. After the commitment changes will be saved.

3.2.35. Use Case: Delete Commits

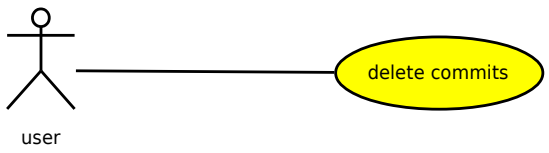


Figure 43. Use case diagram for Delete Commit

Brief Description:

User can delete commits from Smart reminder using this functionality. To delete any commit, login to system, enter Smart Reminder application and select commit in the commits list steps must be successfully accomplished by the user.

Step-by-Step Description:

1.User logins to the system and selects “Smart Reminder” button.
2. User clicks “New Commit” button on the top of Smart Reminder main page and system redirects user to add new commits page.
3. User clicks linked commit area in the commits list.
4. User clicks “Delete Commit” button on the detailed commit page then commit will be deleted.

3.2.36. Use Case: Submit Commits

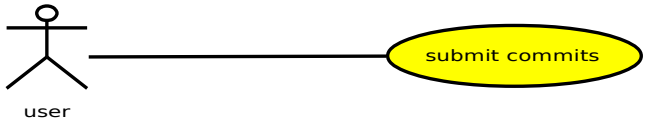


Figure 44. Use case diagram for Submit commit

Brief Description:

Main purpose of the smart Reminder application is to remind commits to user at determined reminder time or randomly. Smart Reminder application asks to user “Did you ….(commit name) today?”. User clicks “Yes” or “No” button in the pop-up screen and commit data will be updated according to this submission. User must have an system account and at least one commit to use this submission reminder part.

Step-by-Step Description:

1. User logins to the system and system must be open the background of the android device.
2. When the achieved reminder time or randomly selected time, system reminds user and asks to
“ Did you ….(commit name) today?” in pop-up screen.
3. User selects “Yes” button or “No” button in this screen.
4. If user clicks “No” button, pop-up screen will be closed. If commit type is free commit, commit data saved as not completed and commit will not be asked again in this day. If commit type is Beeminder commit, commit will be asked again and again during this day randomly. If time is up, data will be saved as not accomplished.
5. If user selects “Yes” button , pop-up screen will be closed and commit data will be saved as yes accomplished and commit will not be asked again in this day. If commit type is Beeminder commit, commit data will be transformed related Beeminder task and task data will be updated.

3.2.37. Use-Case : Create New Tag

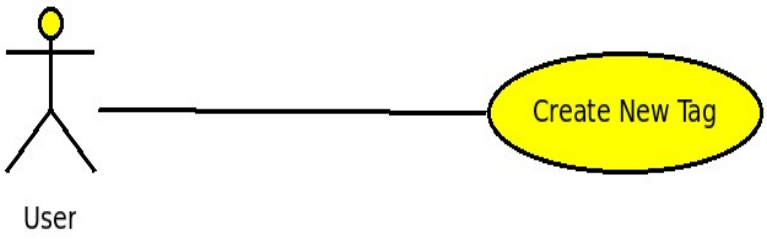


Figure 45. Use case diagram for Create New Tag

Brief Description :

The user can use this case if he successfully logs in the system and clicked on the “Time Tracker” button. He can create a new tag by this use case. The created tag is stored in the database of the application and in the list of tags situated in the “Time Tracker” menu.

Step-by-Step Description:

1. The user select “Time Tracker” button and the system will automatically switch user to selected application layout.
2. In that layout, the system provides a button “Create New Tag”. In case of pressing that button he will be directed to a new layout of application
3. In the opened new layout there are one “Create” button and two empty area waiting to be filled one for “Name Of Tag:” , another for “Duration:”. If the user appropriately fills them and clicks on the button , the new tag will be stored in the database of the application and will be added to the list of tags situated in “Time Tracker “ menu.

3.2.38. Use-Case : Start Tag

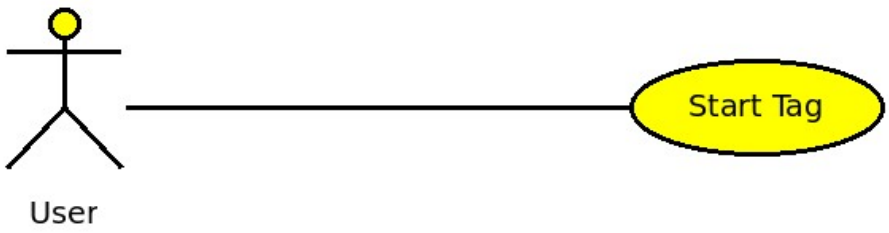


Figure 46. Use case diagram for Start Tag

Brief-Description:

The user can use this case if he successfully logs in the system and clicked on the “Time Tracker” button. By this use case , the time will start and will end automatically after passing duration time determined by user himself while creating the tag.

Step-by-Step Description:

1. The user select “Time Tracker” button and the system will automatically switch user to selected application layout.
2. In that layout,the system provides list of tags where all created tags are arranged. In the left side of the name of the tag created beforehand by the user himself, the user will face with “Start Tag” button.
3. Just after clicking on “Start Tag” button , the time will start and after the duration time determined by the user beforehand when he created the tag it will end.

3.2.39. Use-Case : Delete Tag

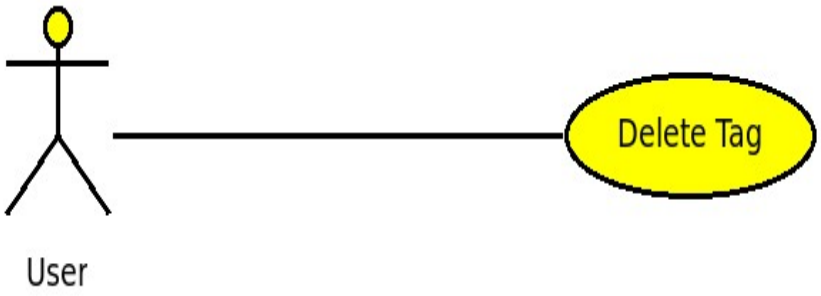


Figure 47. Use case diagram for Delete Tag

Brief-Description:

The user can use this case if he successfully logs in the system and clicked on the "Time Tracker" button. By this use case , the created tag will be removed from both the list of tags and from the database of the application.

Step-by-Step Description:

1. The user select "Time Tracker" button and the system will automatically switch user to selected application layout.
2. In that layout,the system provides list of tags where all created tags are arranged. In the right side of the name of the tag created beforehand by the user himself, the user will face with "Delete Tag" button.
3. Just after clicking on "Delete Tag" button , the tag will be removed from both list of tags and database of the application.

3.2.40. Use-Case : Edit Tag

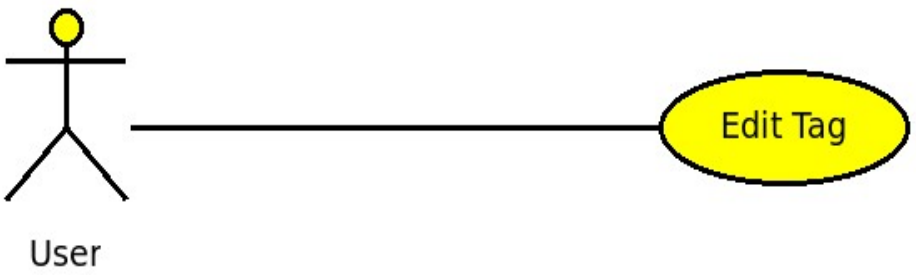


Figure 48. Use case diagram for Edit Tag

Brief-Description:

The user can use this case if he successfully logs in the system and clicked on the “Time Tracker” button. By this use case , the name and the duration of the created tag could be changeable.

Step-by-Step Description:

1. The user select “Time Tracker” button and the system will automatically switch user to selected application layout.
2. In that layout,the system provides list of tags where all created tags are arranged. In the right side of both the “Delete Tag” button and the name of the tag created beforehand by the user himself, the user will face with “Edit Tag” button.
3. If the user clicks on this button , he will be directed to a new layout of application where he will be able to change both Name and Duration of the tag by changing what has been written in blanks “Name Of Tag:” and “Duration:” beforehand and pressing “Save Changes” button.

3.3 Non-functional Requirements

3.3.1 Performance requirement

1. Any data entering to Beeminder with connection over any sub applications will return response not more than max 5 seconds.
2. Any data entering to Database over sub applications will return response not more than max 5 seconds.
3. Since our appearance should be attractive we should lessen our computation count as much as possible.

3.3.2 Design constraint

3.3.2.1 Reliability

1. Our application should show exactly the same results when a user wants to see data about his/her tasks.
2. Mean-time-to-Failure should be at most 12 hours.

3.3.2.2 Availability

1. System should be available 24 hours a day, 7 days a week.
2. In case of any failure, system should recover user and system information by aborting operation.

3.3.2.3 Security

1. Another user should not be able to see a user's data if the user does not want to show.

3.3.2.4 Maintainability

1. All documents should explain everything fully and clearly, so that in the future this application can be understood easily.
2. All codes should be written in a manner that can be understood easily, and it should be as short as possible and should not contain complex expressions.
3. All codes will be written by using object oriented approach
4. When programming and designing this application suit, it is important to follow all news about Android devices and software since it improves so fast.

3.3.2.5 Portability

1. The application suit should support all possible android based devices

4. Data Model and Description

4.1 Data Description

4.1.1 Data Objects

The system has six database tables: Time_Tracker , Smart_Reminder, Schedule_Alarm, Schedule, Habit_Control and Daily_Visited_Websites. Tables contents and details are described below :

Time_Tracker: All tags' information is stored in this table with "tagID" as a table primary key, "tag_name" and "tag_duration" are other keys of the table.

Smart_Reminder : Free commits' information which created by users is collected in this table in terms of "commitID" as a table primary key, "commit_name",commit_reminder_time" and "is_committed".

Schedule_Alarm: Information about all scheduled alarms is stored in Schedule_Alarm table including "taskID" and "tableID" primary keys, "task_name", "alarm_name" , and "recorded_voice_id" other keys.

Schedule : It is dependent table of Schedule_Alarm table. Schedule information is stored this dependent table including "tableID" as a foreign key comes from Schedule_Alarm table, "data","time" and "day" other keys.

Habit_Control : It is collection of all habits' information with "taskID" as a primary key,"task_name", "start_time","task_deadline" and "is_active" keys.

Daily_Visited_Websites: Some required information about daily visited websites is taken this table .It depends Habit_Control table with "has" attribute and it stores some websites information in "websiteID" primary key, ""link", "enterance_time" , "leaving_time" other keys.

4.1.2. Relationships and Complete Data Model

Table 2. Time_Tracker table

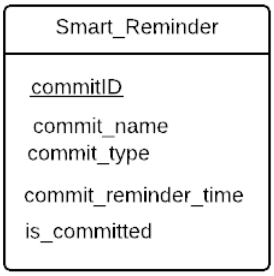


Table 3. Smart_Reminder table

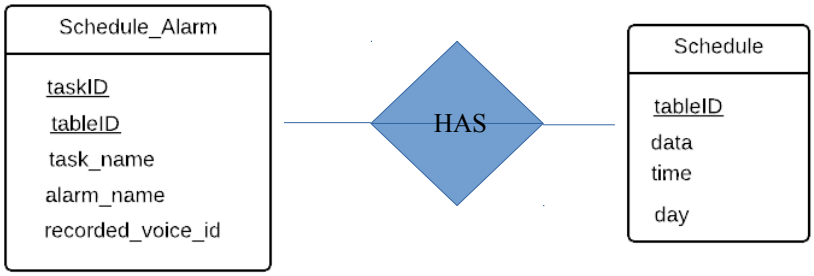


Table 4. Schedule_Alarm & Schedule tables

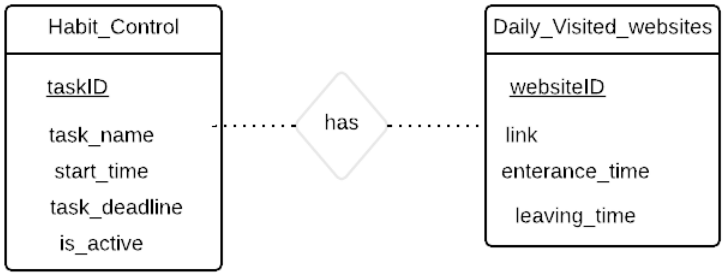
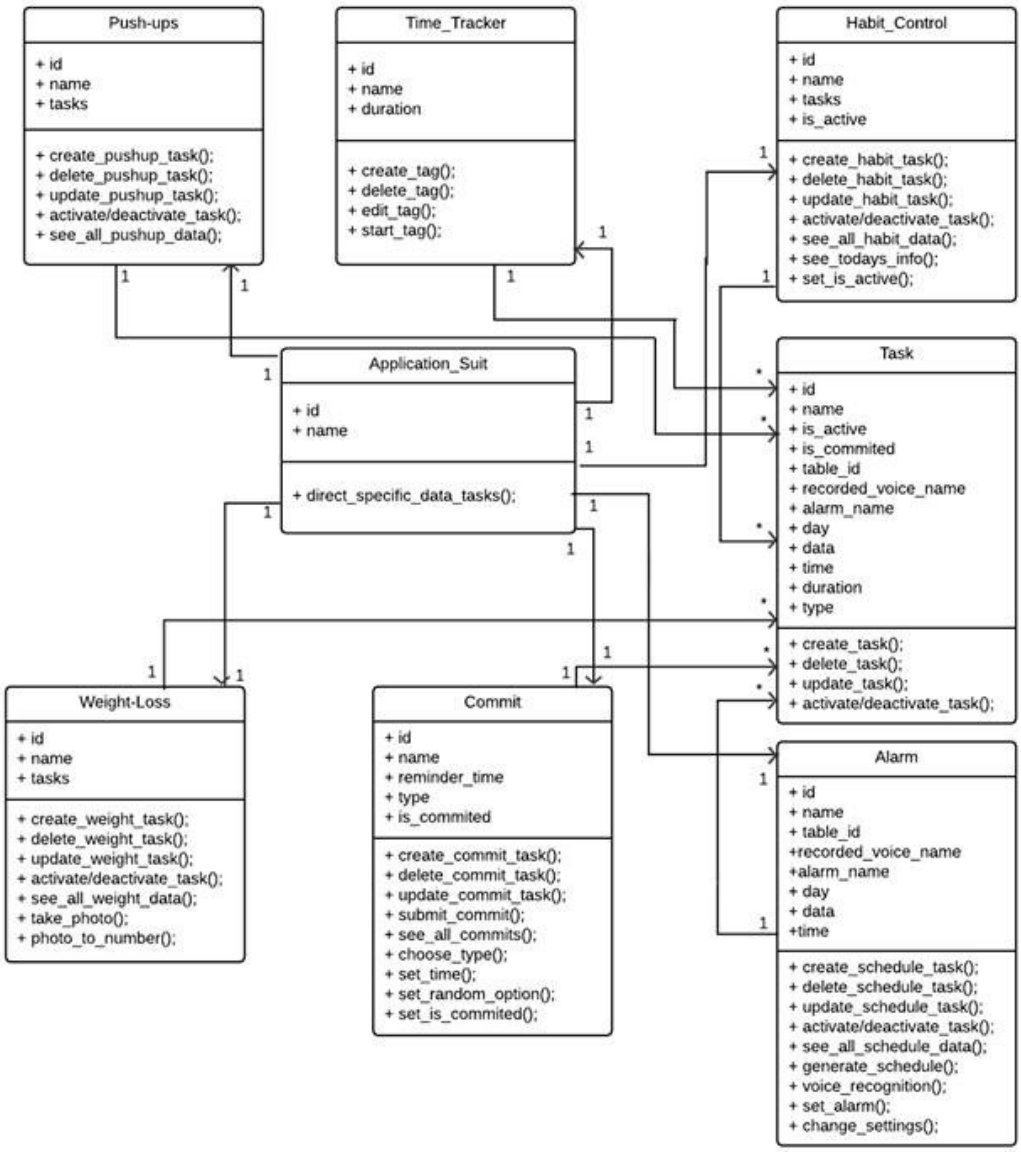


Table 5. Habit_Control & Daily_Visited_websites tables

5 Behavioral Model and Description

Figure 49. Class Diagram



There will be 8 different classes in our project. When the user enters the application, he/she can choose either Push-ups, Time_tracker, Habit_Control, Alarm, Commit or Weight-loss task classes. All of these classes make the desired changes on the Task class. All of these functions are specified in the class diagram. Application suit class is the only way for user to reach other classes. In addition, application suit cannot reach the Task class without reaching one of the other classes, that is, Push-ups, Time_tracker, Habit_Control, Alarm, Commit or Weight-loss classes.

6. Planning

6.1 Team Structure

Our team, ARES , consists of four students. Alihuseyn Gulmammadov (1848282), Songul Abuzar (1678614) , Rufet Eyvazli (1645894), Esragul Korkmaz (1881341). Actually our project consists of from 6 sub applications. We divided 4 of them among the group members and the last 2 will be done together.

6.2 Estimation (Basic Schedule)

Till the end of first semester, as a team we are planning to make the 75 percent facilities available in four sub applications. In the second semester we are planning to complete all the facilities for whole applications suite.

Team Schedule		
Week1	Research about Beeminder	Completed
Week2	Research about opponents	Completed
Week3	Android beginner tutorials	Completed
Week4	Prepared 'User stories & Proposal' Document	Completed
Week5	Android beginner tutorial 2	Completed
Week6	Android beginner tutorial 2	Completed
Week7	Prepared Retrospective Document	Completed
Week8	Android intermediate tutorial	Completed
Week9	Android intermediate tutorial & Project Implementation	Completed
Week10	SRS Document	Completed
Week11	Prepared Retrospective Document 2	Incomplete

Week12	Android advance tutorial & Project Implementation	Incomplete
Week13	Project Implementation	Incomplete
Week14	Prepared Retrospective Document 2	Incomplete
Week15	SDD Document	Incomplete
Week16	Presentation	Incomplete

Table 6. Team Schedule

6.3 Process Model

Since apply any change on the project is to easy. Because as a team, we use 'Scrum Agile' process model. As explanation to scrum agile methods, it has proven to be successful for small sized projects that can be developed by a small team as we are. In this methodology there is 'Scrum master' who arranges all meetings and tracks the backlog of work to be done. During this meetings all team members be aware with any changes if exists and all process details. It is actually means if any problem occurs, the whole change on project will be done easily and efficiently.

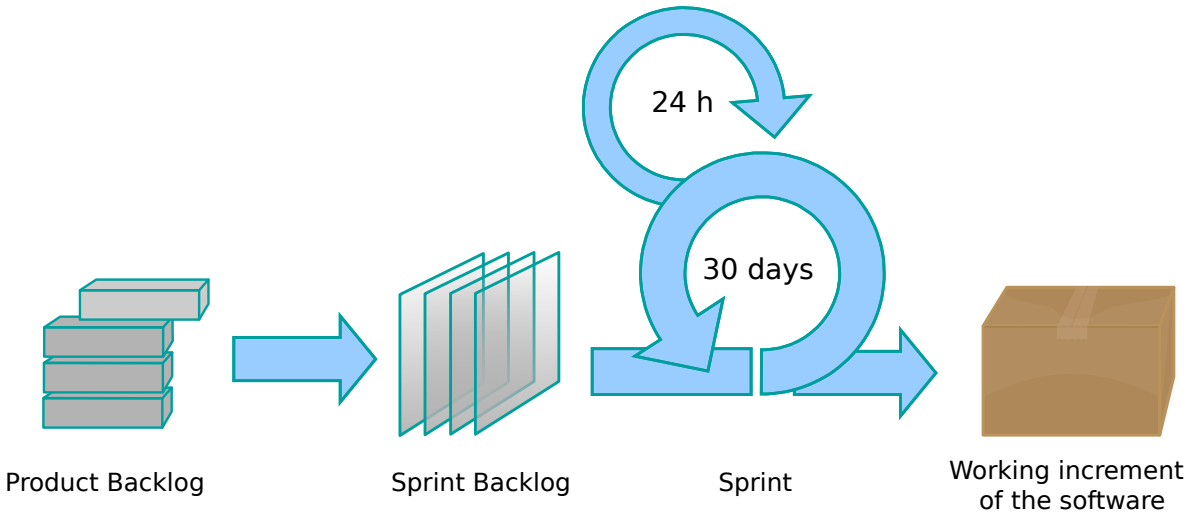


Figure 50. Scrum Agile Process Illustration

7 Conclusion

In conclusion, we have already defined how our applications suite should be and have given its details. For future in implementation part, there maybe some changes and this kind of changes will be noted in the update documents.