SOFTWARE DESIGN DOCUMENT

Version: 1.1
Date: 27.02.2015

Beeminder supported sub applications suite

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

Preface

This document contains the system design information for Android Data Entry Application Suite for Beeminder Project. The document is prepared according to the “IEEE Standard for Information Technology –Systems Design –Software Design Descriptions –IEEE Std 1016 – 2009”. This Software Design Documentation provides wide description of all the system design and views of Android Data Entry Application Suite for Beeminder Project.

The first section of this document includes purpose, scope, intended audience of the project.

The second section of this documentation includes definitions which are abbreviations and acronyms.

The third section of the document explains conceptual model of software design for descriptions.

The fourth section of this documentation widely explains design description information content.

The fifth section of the document give enlarged information about viewpoints used in documentation process.

Change History

Version Number	Date	Brief Description
1.0	04.01.2015	Original
1.1	27.02.2015	The equality between the methods used in class diagram and in view diagram are supplied. Explanation on methods are extended.

Table of Contents

Tables & Figures

1. Introduction

1.1. 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

1.2. Purpose

In this document, design feature of our Beeminder application suite will be covered. When covering this feature, IEEE 1016-2009 Recommended Practice for Software Design Descriptions (SDD) will be considered as a sample to guide us.

This document will firstly give some important definitions in section 2, then explain Conceptual model for software design descriptions and Design description information content at sections 3 and 4 , and finally in section 5, the viewpoints of the design will be covered.

1.3. Intended audience

The audience for the SDD includes the project management, the system architects (i.e. , the developers who participate in the system design), and the developers who design and implement each subsystem.

This Software Requirements document is intended for:

- Developers can use this document in order to understand the software easily and in this way they can easily improve the features or add new functionalities to the system.
- Testers can use this document in order to find some bugs for their testing strategy. Generally, bugs are easy to find by using software requirements document.
- Users can also use this document to understand what they can do with this conference management software and how they can use it.

2. Definitions

Definitions, acronyms, and abbreviations	Definition
SDD	Software Design Description
SRS	Software Requirement Specifications
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

Table 1. Table of Definitions & abbreviations

3. Conceptual model for software design descriptions

In this section, we will present a conceptual model for the SDD. This conceptual model mainly explains the context in which SDD is prepared and how it will be used by the stakeholders and developers. Basic terms, concepts and context of SDD will be given in this part.

3.1. Software design in context

The Easy Data Entry Applications Suite for Beeminder project is an Android application suite project and we are supposed to implement 6 different user friendly goal-tracking applications for this suite. The main aim of these quantified self applications is to supply reliable and easy data entry for Beeminder. The required software products and their usage areas over whole project are described as below:

●The applications suite will 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.

3.2. Software design descriptions within the life cycle

3.2.1 Influences on SDD preparation

Our System Requirements Specifications(SRS) is our basic guide for our SDD progress. When designing our project, our team focuses on the functional and non-functional requirements of our Android application suite project.

3.2.2 Influences on software life cycle products

During our progress, we have changed some features of our project, according to new requirements and constraints that we have learned after our SRS report. In addition, in this progress, when the group decides to change a property for a better usage and appearance, we have replaced this property with the new one.

Our test procedure will be done through the guide of SDD. When writing this document, our group considered all of the requirements for a better testing.

3.2.3 Design verification and design role in validation

In order to determine whether a project meets all the of its requirements, verification method is used. Though, validation method is used for proving that a project meets the requirements of a specific usage. When doing our verification and validation tests, our team focused on the requirements, that we have searched and decided before. For every requirement, we are planning to apply at least one test case and making all of the test cases as independent as possible. These configured test cases will cause more time spending on testing phase but can help to find out the hidden errors.

4. Design description information content

4.1. Introduction

The required contents of an SDD are as follows:

1. Identification of the SDD
2. Identified design stakeholders identified design concerns
3. Selected design viewpoints, each with type definitions of its allowed design elements and design languages
4. Design views
5. Design overlays
6. Design rationale

4.2. SDD Identification:

At the end of the second semester all Beeminder users will get introduced with new functionalities of the Beeminder application in the devices which are Android OS platform based. Moreover , what is the plan for that time is to supply useful, simple and user-friendly interface for the users of the application. The project with all new functionalities of the application will reduce the number of questions starting with the word “WHY?”.

4.3. Design stakeholders and their concerns

The main concerns of the stakeholders are security and efficiency of the application. The main reason is that their accounts and passwords are used in this project.

4.4. Design views

Unified Modeling Language (UML) 5.02 is used for graphical representations of viewpoints in “Android Data Entry Application Suite for Beeminder” Project in Diagrams. For user interface viewpoint part, Android User Interface creator in the link https://www.lucidchart.com/documents/ was used.

4.5. Design viewpoints

A design viewpoint addresses a different perspective to be focused on to effectively encompass requirements that have been previously created and to identify the stakeholders as to which these requirements are relevant. And five viewpoints is used during documentation of SDD which each of them refers the range of design concerns that have been recognized.

4.6. Design elements

Our design entities for each application can be defined as follows :

Push_ups : This application is specialized on push-up activity. User can count and save his/her push-up count easily, by using this application. In addition, committing this count to Beeminder is easier with this application.

Time_Tracker : This application is specialized on keeping track of elapsed time when doing an activity. While doing a task, user can decide time_interval and see how much time elapsed. In background, this task completion time will be sent to related goal on Beeminder as a new data entry.

Weight_loss : This application is specialized on weight control of the user. User can commit his/her weight by only capturing a photo of his/her current weight value on the scale. Then, committing this value is occurred automatically by using photo to number recognizer.

Smart_Reminder : This application is specialized on deciding a schedule in a user specified time. User can keep track of his/her tasks and be warned about doing them. In this way, forgetting to do something is prevented by this application.

Habit_Control : This application is specialized on excessive Internet usage time. While the user starts to use the Internet, timekeeper will start to keep track of user's time usage on Internet. At the end of the day, it commits total amount of Internet usage time to Beeminder automatically.

Schedule_Alarm : This application is specialized on guaranteeing a regular task's completion. That is, user can arrange his/her schedule and the according to the schedule user will be alerted with alarm system of application. The alarm system will be deactivated with voice recognition to guarantee that user is awake. The attendance according to schedule will be edited to Beeminder automatically.

4.6.1 Design entities

The list of design entities of our project is as follows :

Application_suite : Application suite class keeps all of the six applications, mentioned below, together.

push_ups : This class is specialized on simplifying usage of Beeminder on Android OS, only for push_up tasks.

time_tracker : This class is specialized on simplifying usage of Beeminder on Android OS, only for the users who wants to keep track of the time elapsed, while doing a task.

weight_loss : This class is specialized on simplifying usage of Beeminder on Android OS, only for the users who wants to commit their weight value easier.

smart_reminder : This class is specialized on simplifying usage of Beeminder on Android OS, only for the users who wants to keep track of different tasks to be done and be warned to complete them.

habit_control : This class is specialized on simplifying usage of Beeminder on Android OS, only for the users who wants to keep track of the time elapsed when using the Internet, and committing it to Beeminder automatically.

schedule_alarm : This class is specialized on simplifying usage of Beeminder on Android OS, only for the users who wants to be forced to complete a task continuously, when he/she did not complete it, in time.

4.6.2 Design attributes

push_ups :

id : unique id of this application.
name: name of this application
tasks : tasks of the user which have the type of push_ups activity

time_tracker :

id : unique id of this application
name: name of this application
duration : specified time interval of the task

weight_loss :

id : unique id of this application
name: name of this application
tasks : tasks of the user which have the type of push_ups activity

smart_reminder :

id : unique id of this application
name: name of this application
reminder_time : defines the time-interval to complete the tasks
type : defines which task to be completed
is_committed : defines whether this task is committed or not

habit_control :

id : unique id of this application
name: name of this application
tasks : tasks of the user which have the type of push_ups activity
is_active : defines whether the task is currently active or not

schedule_alarm :

id : unique id of this application
name: name of this application
recorded_voice_name : defines the voice name of tasks
table_id : id of table
alarm_name : name of different alarms
day : specified day of tasks
date : specified date of tasks
time : specified time of tasks

4.6.3 Design relationships

Functionally :

All of the six applications above is specialized on a unique activity task, although their aim is the same, user-friendly interface and easier data committing to Beeminder. All of the applications, push_ups, time_tracker, weight_loss, smart_reminder, habit_control and schedule_alarm, is integrated in a single application suite.

Non-functionally :

We are planning to design the user interface of all applications in a way that they look almost the same. However, since each application requires some different features, we will add this feature without changing the basic appearance much, for a user-friendly approach.

4.6.4 Design constraints

In our project, there will be two main constraints. First constraint is, appearance concerns. That is, since we are designing an application suite, all of the six applications, mentioned above, should have similar user interface for an user-friendly approach.

Second constraint is, about Beeminder format. That is, we are trying to develop an application suite for an existing project, namely Beeminder. That is why, we should keep some data in device's database, since Beeminder only keeps general data information. However, our project is specialized on simplifying some common usages, so that it needs more saved data for a better usage.

4.7 Design overlays

We have changed our appearance considerations, so that all of the 6 applications have similar appearance. In addition, we were thinking about connecting to Beeminder ourselves, however, we have learned that our associated instructor, Uluc Saranli, has written them before, so there is a library at Github to be used. Since we are doing our applications one after another, we could only finished our push_ups application. New design integration on push_ups application will be usable for all suite applications to get better user interface. Creativity on this issue is required to come up final result.

4.8 Design rationale

The main aim is decided to maintain Beeminder usage over all suite applications. Since the beginning of this semester, as a team, we have been using Beeminder to learn its concepts better. In addition, for a better approach to our 6 applications, we are getting some advice from our friends, who can use this application. In this way, we can see people's needs in more point of views.

4.9 Design languages

We are using UML to show our design approaches about this project.

5. Design viewpoints

5.1. Introduction

In this section, five main design viewpoints will be explained widely. During the explanation of these given viewpoints in below, UML diagrams will be used for clarification. The viewpoints are as below:

1. Context Viewpoint
2. Logical Viewpoint
3. User Interface Viewpoint
4. Dependency Viewpoint
5. Interaction Viewpoint

5.2. Context viewpoint

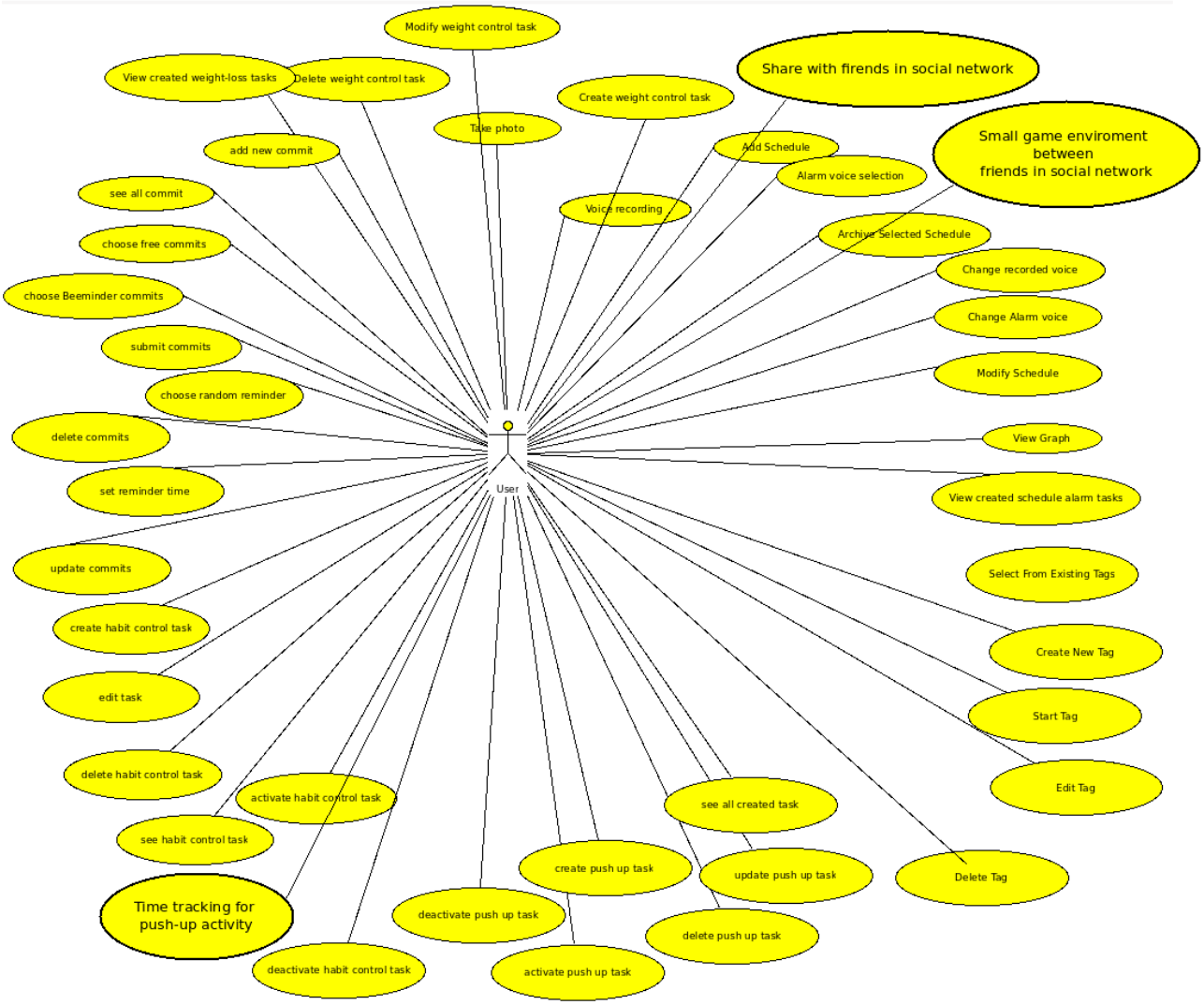

Figure 1: Use Case Diagram

5.1.1. Design Concern

Concern of the context viewpoint is to determine functionalities offered for the user. In terms of the efficiency of the user features, it is important to give explicit definition to these features.

5.1.2. Design Elements

Detailed information for supplied functionalities was stated in Software Requirement Specification document. In Software Design Description document the added functionalities will be mentioned.

5.1.2.1. Use Case: Share with friends in social network

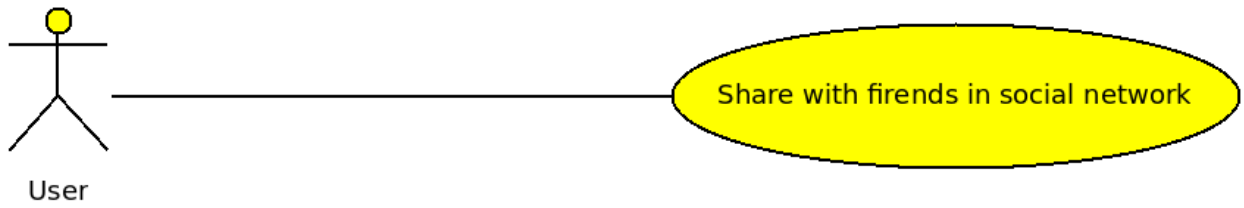

Figure 2: Share with friends in social network Use Case diagram

Brief Description:
User can use this functionality after successfully login. This functionality gives opportunity to share his success with friends in social networking websites. This functionality will be applied to all sub applications in the suite.

Step-by-step Description:

1. User login to system and select any application among the six applications.
2. Starts doing required tasks. For instance, in push-up application does some push-ups.
3. After completing task, a button for share will be appeared.
4. Click on the share button will share required information in user profile related social networking website with friends in same format as for other applications.

5.1.2.2. Use Case: Track and comparison between friends in social network

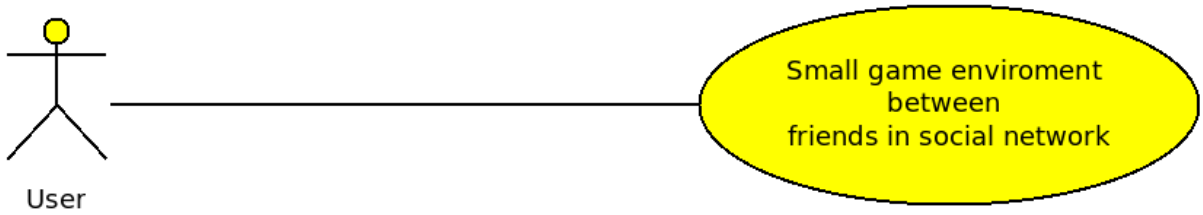

Figure 3: Track and comparison between friends in social network Use Case diagram

Brief Description:
User can use this functionality after successfully login. This functionality gives opportunity to user track his friends who are doing same activities. The provided list which is showing friend's point will be sorted in ascending order by using their points on tasks. . This environment creates small contest between users.

Step-by-step Description:

1. User login to system and select any application among the six applications.
2. By click on friend track button redirect user to new layout.
3. In this new layout the list of friends including user will be shown in sorted order according to points.

5.1.2.3. Use Case: Time tracking for push-up activity

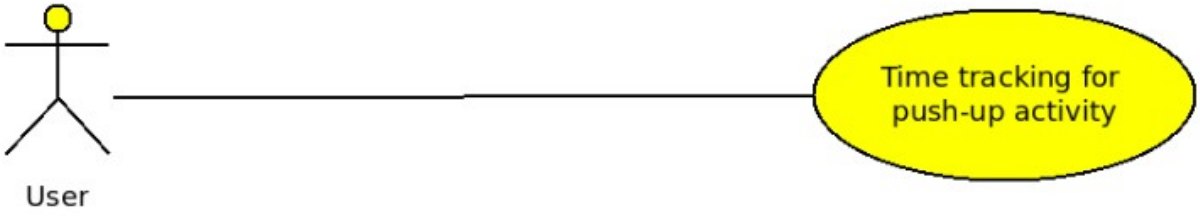

Figure 4: Time tracking for push-up activity Use Case diagram

Brief Description:
User can use this functionality after successfully login. This functionality gives opportunity to user to track the push-up time. Which means user can see his ability and improvement for a specific time interval.

Step-by-step Description:

1. User login to system and select any push-up application.
2. In first layout, user enter the count down time for push-up task. Otherwise it will disabled.
3. In push-up layout the application will start to count until given time finished.

5.3 Logical viewpoint

This viewpoint aims to show the key abstractions such as classes and interactions among them. UML Class diagram is provided for this aim which can be seen as below:

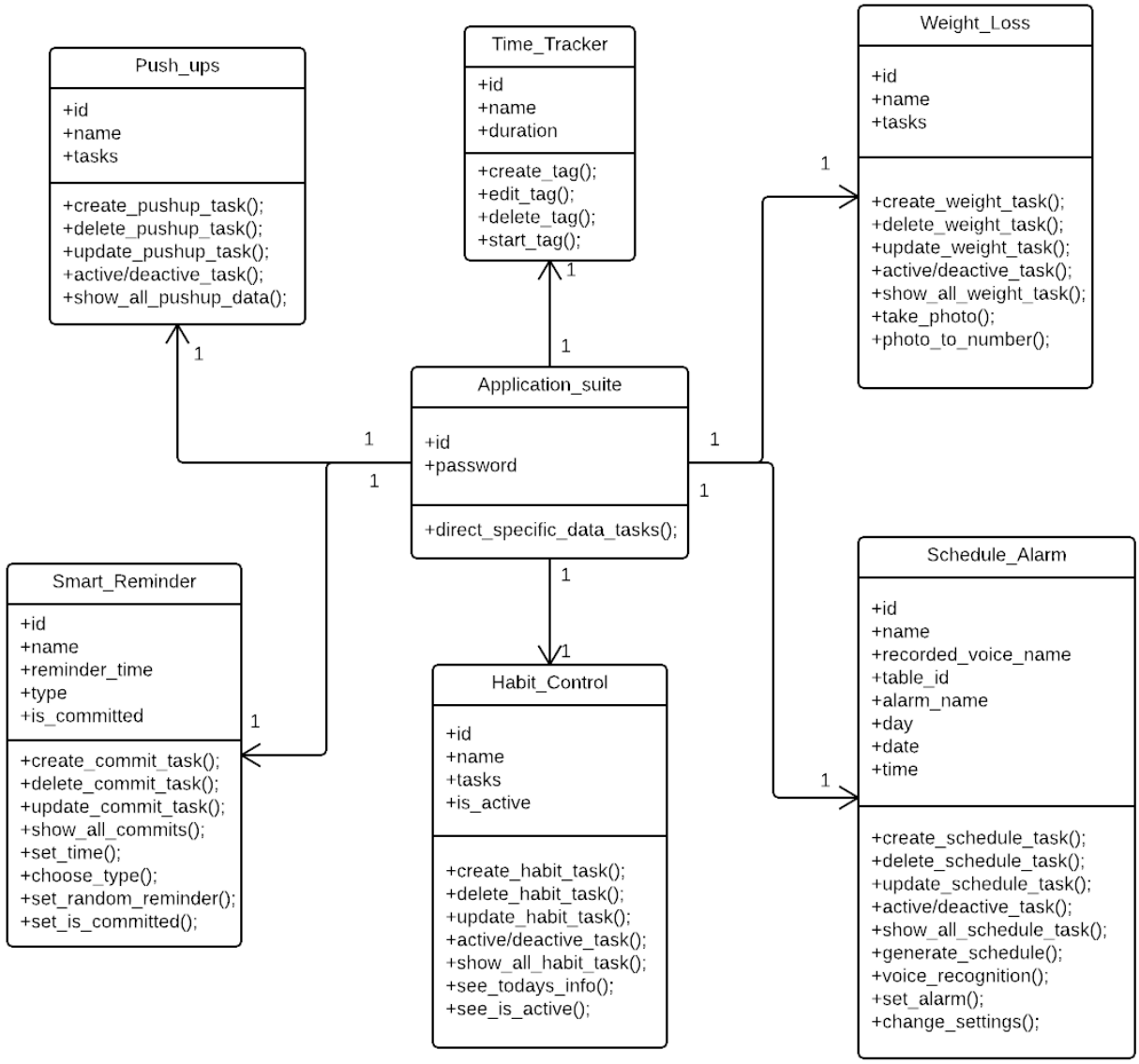

Figure 5: Class Diagram

There will be 7 different main classes in our project in terms of Application_suit, Push-ups, Time_tracker, Habit_Control, Schedule_Alarm, Smart_Reminder and Weight-loss . We will describe in detail as below sections.

5.3.1. Application_suite Class

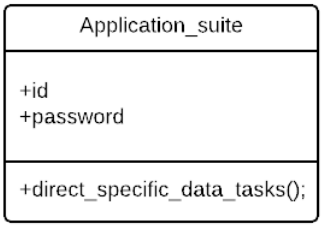

Figure 6: Application suite

When the user enters the application suite with using their Beeminder username and password, he/she will be faced with Application_suite class. This class is the only way for user to reach other classes. User only can reach other suite classes after that it enters the application suite class.

5.3.1.1. Fields

- **username :** This is a string value that is unique for every user. This is selected by the Beeminder user which have Beeminder user account.
- **password :** Every Beeminder user has his/her own password that is selected by him/her. It is also string value can includes characters and digits.

5.3.1.2. Methods

- **direct_specific_data_tasks :** This method is called when user wants to enter the application suite with using their Beeminder username and password. After user clicks login button, this method will be called by the system and s/he will be directed application suite main page.

5.3.2. Push_ups Class

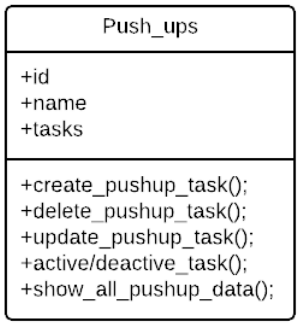

Figure 7: Push ups

In the application suite main page, user can select 6 different application classes. Push_ups class is created for the athletic users or wants to be athletic. It provides push up counter for user and Beeminder. Also, time counter and push up steps are another functionalities of this classes. There is no direct connection between push_ups class and other suite classes apart from Application_suite class.

5.3.2.1. Fields

- **id :** This is an integer value that is unique for every push ups tasks. This id is chosen by the system by auto incrementation.
- **name :** This string value is unique for ever push ups tasks.
- **tasks :** tasks field is a string list value. Its length changes with push ups task number.

5.3.2.2. Methods

- **create_pushup_task :** This method is called when user wants to create new push-ups task and store in the Beeminder database. User can use this method with clicking Create New button at top of Push Ups application main page and s/he can create new push ups task and store it Beeminder database after entering necessary fields in this page .
- **delete_pushup_task :** When user wants to delete any push ups task from Beeminder database, s/he selects any task from push ups tasks list and clicks delete button at the right top of the this page. Then, this method is called and task is deleted.
- **update_pushup_task :** It provides updating task Push ups task information stored in Beeminder database. With selecting any tasks from tasks list, user can change features of task with the help of this method. This method is called when user wants to see all task data with selecting any task from tasks list and clicking edit button on left top of the push ups application main page.
- **active/deactive_task :** This method used for sleeping selected task for a given period.
- **show_all_pushup_data :** Listing all Push ups tasks operation is provided by this method. When user clicks any list element on the task list, it will be called this method and user can see all about data information regardless of this method.

5.3.3. Weight_Loss Class

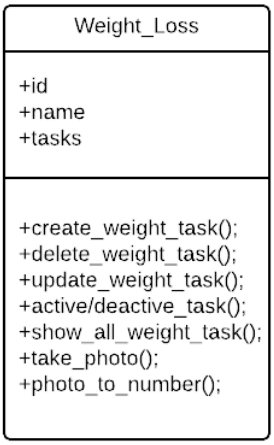

Figure 8: Weight-Loss

Weight_Loss class is created to keep full control on weight loss data. The user's weight on scale will provide data for Beeminder with the help of optical character recognition libraries of Android. User can reach this class with entering Application_suite class and then selecting weight loss application. Like push_ups class, it is only directly connected with application_suite class.

5.3.3.1. Fields

- **name :** This string value is unique for every Weight Loss tasks.
- **tasks :** Tasks field is an string list value. Its length changes with Weight Loss task number.
- **id :** This is an integer value that is unique for every weight loss tasks. This id is assigned by the system auto incrementation.

5.3.3.2. Methods

- **create_weight_task :** When user wants to create new Weight Loss tasks and save it Beeminder database, this method will be called. When the Create New button is clicked, system calls this method and with filling necessary task fields and clicking Add button, creating new Weight Loss task functionality will be completed.
- **delete_weight_task :** This method is called when user wants to delete any task from Beeminder database. With selecting desired tasks from list and then clicking delete button at the right top of the page, deleting weight loss progress will be completed.
- **update_weight_task :** Beeminder Weight Loss tasks informations will be modified with using this method. System calls this method when user selects a task from task list and clicks edit button at left top of the weight loss main page. In the opened layout, the provided features can be reedited by user and will be completed by click on the edit button.
- **active/deactive_task :** This method used for sleeping selected task for a given period.
- **show_all_weight_data :** To list all Weight Loss task name stored in Beeminder database is provided by this method.
- **take_photo :** This method is called when user wants to take a photo of his weight while is standing on scales. Taken photo supplies data entry for weight loss application with optical character recognition.
- **photo_to_number :** To convert image to integer, user must called this method. Taken photo while is standing on scales will be converted to character and user selects necessary numbers from these characters and application supplies weight digit for user.

5.3.4. Smart_Reminder Class

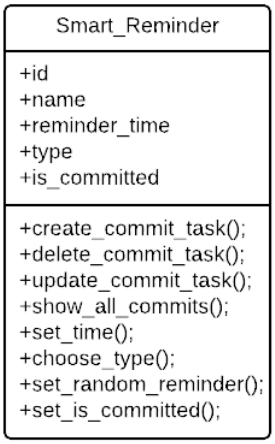

Figure 9: Smart Reminder

Smart Reminder class is used to remind edited task by user in time. During of the day, the questions about task will be asked to user to identify the task completion. This application prevents procrastination. It is not only usable to check user for a specific task. It also help user to keep to-do list in up to date.

5.3.4.1. Fields

- **id :** This integer value is assigned by the system as a unique task identification number for every Smart Reminder application tasks.
- **name :** this is a string value and it is unique for every smart reminder task and it is assigned by user.
- **type :** It is a string value and every smart reminder task is in free type or Beeminder type. It is also assigned by user.
- **reminder_time :** It is a date time value includes date and time for task reminder and assigned by user.
- **is_committed :** It is boolean value and it will be used for checking whether task completed or not. Initially, for all created tasks, this value will be false. After committing task the default value will be changed from false to true by the system.

5.3.4.2. Methods

- **create_commit_task :** This method is called when user wants to create new Smart Reminder task.
- **delete_commit_task :** It is used for deleting Smart Reminder tasks from Beeminder database. System calls this method when user selects one or more task from task list and clicks delete button at the right top of the smart reminder main page.
- **update_commit_task :** Owing to this method, user can change any task information stored in the Beeminder database. System calls this method when user selects a task from task list

and clicks edit button at left top of the smart reminder main page. After the changes on desired field or fields, with clicking edit button, updating task information progress will be completed.

- **show_all_commits :** It is used for showing all Smart Reminder tasks. As soon as entering the smart Reminder task, this method will be called by the system and user can see all commits in linked list format at the smart reminder main page.
- **set_time :** When user desires add new Smart Reminder task or edit reminder type, it will be called by the system. The method will supply reminder time.
- **set_random_reminder :** Reminder time value assigned by the system as random time value for selected with calling this method.
- **choose_type :** When adding new tasks or updating tasks information, this method will be called. User can select two type of tasks by calling this method. These two types are free task which means has no any connection or interaction with Beeminder and Beeminder connected task.
- **set_is _committed :** It is called automatically by the system, when user commits the reminded tasks. If the task is committed, reminder will be disabled for this committed task during that day.

5.3.5. Time_Tracker Class

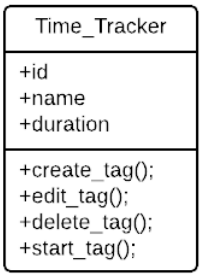

Figure 10: Time Tracker

This class provides specific time interval for user to focus on specified task. It keeps time for different work facilities and make to focus on current work for selected time interval. It gives a duration for Beeminder goals. It is reachable with only application suite class and there is no direct relation between Time_Tracker class and other application suite classes.

5.3.5.1. Fields

- **name :** This is a string value and it is unique for every Time Tracker task.
- **id :** This is an auto incremented integer value assigned by the system for every task as a unique identification number.
- **duration :** This value is a time value and it will be assigned by user for every Time Tracker task.

5.3.5.2. Methods

- **create_tag :** This method is called when user wants to create a new tag. After entering time tracker application, with clicking Create New button and filling necessary fields this method will be called and completed.
- **delete_tag :** When user wants to delete any Time Tracker task on database, this method will be called. User selects any tasks from the tag list and clicks delete button at the right top of the Time Tracker main page, then this method is called and tag is deleted from system.
- **edit_tag :** It is used for changing any Time Tracker task information from Beeminder database. System calls this method when user selects a task from task list and clicks edit button at left top of the time tracker main page. With changing some fields and clicking edit button, updating task information progress will be completed.
- **start_tag :** This method is called when user clicks the start button on the screen timer. With calling this method, timer will be start.

5.3.6. Schedule_Alarm Class

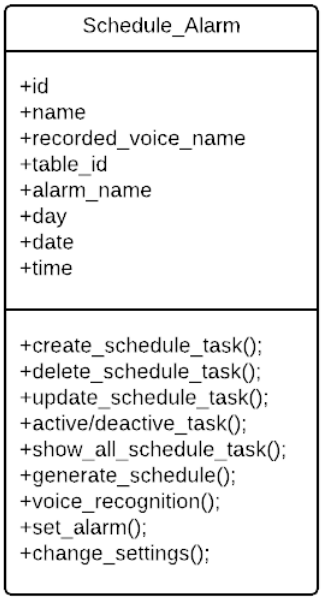

Figure 11 Schedule Alarm

Schedule_Alarm provides voice recognized alarm system and keeps track of attendance for added schedule. It is only directly connected with Application_suite class and there is no directed relation with other classes.

5.3.6.1. Fields

- **id :** This is an integer value that is unique for every Schedule Alarm task. This id is assigned by the system auto incrementation.
- **name :** This string value is unique for every schedule alarm task and it provides task name.
- recorded_voice_name : This is specified name for every recorded user voice.
- **table_id :** This is an identification integer value for every task and it is assigned by the system automatically for every tasks.
- alarm_name : this is a string value for every alarm and it is unique for every task.
- **day :** this is string value in terms of days name intervals.
- **date :** it is date value for every task alarm date.
- **time :** it is time value for every task alarm time.

5.3.6.2. Methods

- **create_schedule_task :** User can add new schedule item for Schedule Alarm application with calling this method. The created schedule will be kept in the database of application.
- **delete_schedule_task :** It is used for deleting any Schedule Alarm task stored in he Beeminder database. This method is called when user wants to delete one or more schedule tasks from tasks list. With selecting desired tasks from list and then clicking delete button at right top of the page, deleting task progress will be completed.
- **update_schedule_task :** Owing to this method, any Schedule Alarm task information can be modified and updated from Beeminder database. When user wants to change schedule alarm task informations, this method will be called. User selects a task from task list and clicks edit button at left top of the schedule alarm main page. After updating provided fields, with clicking edit button updating task information progress will be completed.
- **active/deactive_task :** This method used for sleeping selected task for a given time interval.
- **show_all_schedule_tasks :** User can see all Schedule tasks with calling this method. When calling this method, all Schedule Alarm task stored in Beeminder database will be listed on the screen.
- **generate_schedule :** This method is called when user wants to generate scheduler for alarms.
- **voice_recognition :** User's voice is recorded with this method. This recorded voice will be used to deactivate alarm system with voice recognition.
- **set_alarm :** When adding new schedule alarm task or updating task information, this method will be called. It provides to set alarm at desired date and time.
- **change_settings :** It is used for changing schedule alarm application settings and it will be updated by user.

5.3.7. Habit_Control Class

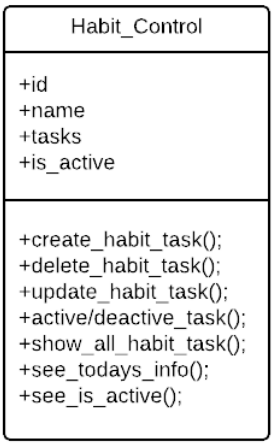

Figure 12: Habit Control

Habit_Control try to make restriction on Internet usage with working background and provide spent time as a new data entry. It provides a spend time duration for user and system specific websites. This class only related with application suite class.

5.3.7.1. Fields

- **id :** This is an integer value that is unique for every Habit Control task. This id is assigned by the system with auto incrementation.
- **name :** This string value is unique for ever habit control tasks.
- **task :** Tasks field is an string list value. Its length changes with weight loss task number.
- **is_active :** It is a boolean value and give information about the task is active or not.

5.2.7.2. Methods

- **create_habit_task :** It is used for creating new Habit Control task. After created new Habit Control task, it will be stored in Beeminder database.
- **delete_habit_task :** This method is called when user wants to delete any Habit Control task stored in Beeminder database. With selecting one or more task from list and then clicking delete button at right top of the page, deleting habit control progress will be completed.
- **update_habit_task :** All Habit Control task information can be modifiable with calling this

method. System calls this method when user selects a task from task list and clicks edit button at left top of the Habit Control main page.

- **show_all_habit_tasks :** It is used for listing all Habit Control tasks.This method is called when user wants to see all habit control tasks.
- **see_todays_info :** This method is called when user wants to see INTERNET usage information during that day.
- **see_is_active :** To show the any Habit Control task is active or not, this method will be called.

5.4. Dependency viewpoint

Our dependency diagram can be shown in the figure below. As specified in the figure, our application is in action with both main Beeminder Server and the user. After entering the user information, our application connects to main Beeminder server to get detailed information about the user, if the user name and password is correct, otherwise it warns the user to correct the user name or password.

If there is any Internet connection and access to Beeminder server, application will directly get the goal names from the server in order to direct the user to the welcome page. When the user selects a goal, our application again connects to main Beeminder server and gets detailed information about the task. According to which property of our application that the user uses, our application gets some information from the database of the device, which consists of the information that Beeminder does not keep, but our application needs.

5.4.1. Design concerns

In the perspective of the programming language and programming environment, our team uses Java as the programming language and Eclipse with ADT as the programming environment.

When connecting to the main server of Beeminder, we use the supported library of Beeminder written by Uluc Saranli, and can be found at Github. It has already written to connect to the Beeminder server from an Android OS and tested. In this way, we do not need to consider the bottleneck or isolating entities problems when connecting.

5.4.2. Design Elements

Our design entities here is the user, per-supplied Android - Beeminder connection libraries obtained from Github, main Beeminder server, application's database and the application itself.

User uses the application. Application requires information from device's database and/or main Beeminder server according to user's requirement. When getting information from the main Beeminder server, libraries from Github provides with all connection operations. All attributes can be seen in the following table for each entity.

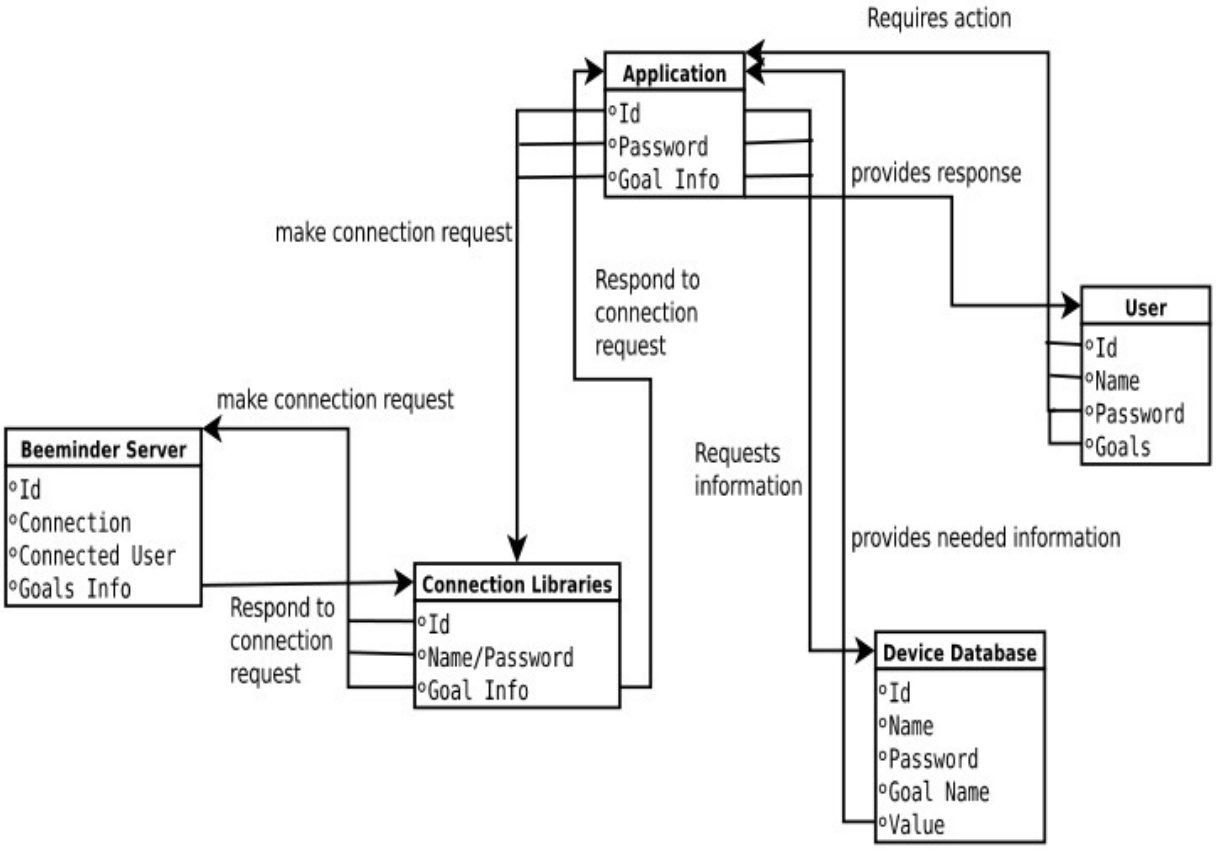

Figure 13: Dependency Diagram with Attributes

5.5. User Interface Viewpoint

In the SRS documentation, some of the user interface mock-ups were arranged so as to get familiar with what the project will look like at the end of the second semester. In this Documentation, some additions to the user interfaces and some explanatory mock-ups were used in order to show all of the steps starting from LOGIN. This will help understanding the project visually.

5.5.1. The Login User Interface

Figure 14: Login User Interface

Because of the fact that the Beeminder application supplies an account for each user , the user will type his/her login and password informations for accessing his/her account, in the login user Interface .

5.5.2. Functionalities List Interface

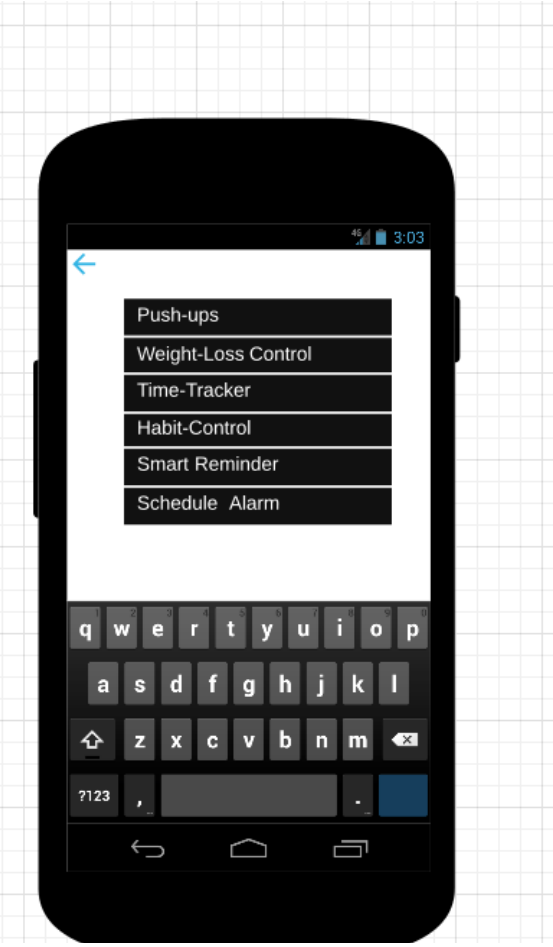

Figure 15: Functionalities List

After successfully login operation, the user will face with all of the functionalities listed above in the figure .All listed items are clickable.

5.5.2.1 In case of clicking on Push-ups button :

The user will be directed to the **push_ups** application page. In this page, there will be a **list of tasks , add**, **edit** and **delete** buttons as figure in the below. In case of clicking on **add** button , the user can add a new task and the added task will be placed in the **list of tasks** and stored in the database of the Beeminder application. If the user clicks on the **edit** button, he will be able to change some properties of the task added before. In case of clicking **delete** buttons any selected tag might be deleted from both database and the **list of tags**. In case of clicking on any task name in the **list of tags** , the user will be directed to the **Start The Push-ups Count Page**

Figure 16: Push-Ups Application Page

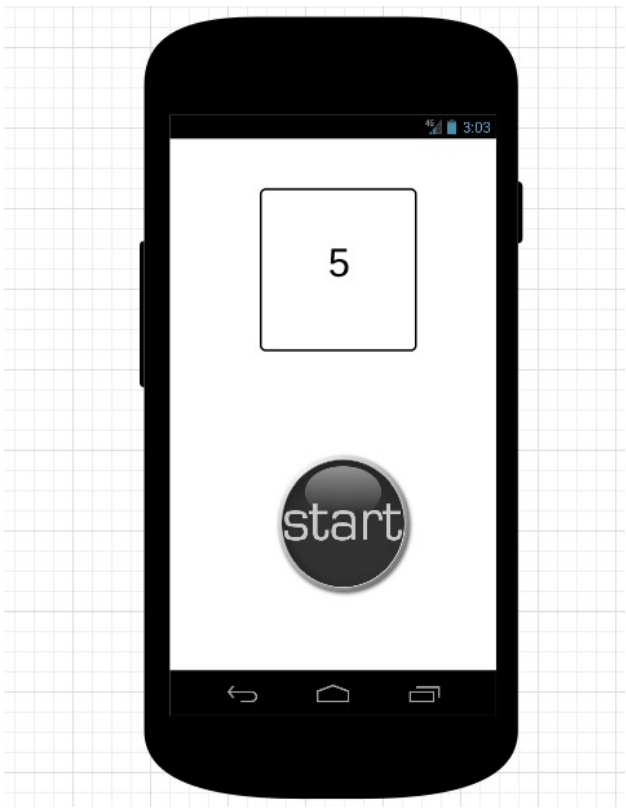

Figure 17: Start Push-Up Count Page

5.5.2.2. In case of clicking Weight-Loss Control button :

The user will be directed to the **weight-loss control page** and for request to take photo of the digital scales will redirect user to a layout which is shown in the Figure 18. The data of the scales will be pulled as an input to the Beeminder application like in the figure below. This will help the user enter the data to his/her task account from reliable and easy way.

Figure 18: Weight-Loss Control Page

5.5.2.3. In case of clicking Time-Tracker button :

The user will be directed to the page where there are **list of tags , add new tag, delete tag, edit tag** and **start** buttons. By the help of **add** button the user will be able to add new tag to both database of Beeminder application and in the **list of tags** of the page. **Delete** button, on the other hand , removes them from both sides. **Edit** button helps making some changes on the previously added tag . The tags are listed under the **list of tags** as showed in the figure. If the user clicks on the **start** button , the time starts from 0 and goes until it reaches to the previously decided (when the tag was created) time. By the help of this new functionality the user will be able to control himself/herself in that time interval .

Figure 19: Time-Tracker Page

5.5.2.4. In case of clicking Habit-Control button :

The user will be directed to the Habit-Control page. In this page he/she will face with a question such as Do you ACTIVATE or DEACTIVATE as shown in the Figure 20 . Initially, ACTIVATE is selected. As long as the activate button is selected , the user is responsible from doing all created tasks. However in case of selection of DEACTIVATE button he/she is not responsible from any tasks. Deactivate button means that “I am not feeling okey to handle the tasks”.This is like freezing everything for one day. However, the user can not always do this. For this, he/she will have a limitations. Those limitation is for keeping the responsibilities of the user under control.

Figure 20: Habit-Control page

5.5.2.5. In case of clicking Smart Reminder button :

The user will be directed to the Smart Reminder page .In this page, the user will select any of the task which he/she wants to be reminded about. According to the selection , the notification will come to him often. For instance, if he/she chooses **push-ups task** to be reminded about. A notification like in the figure, will come to his device so as to remind him/her about what he/she should do. If he/she has already done the task, he/she clicks YES otherwise NO button. In case of choosing pressing NO button , the notification will take place again and again until YES button is clicked.

Figure 21:Smart Reminder Notification

5.5.2.6. In case of clicking Schedule Alarm Button :

The user will be directed to the Schedule Alarm page. In this page , the user will adjust an alarm task. According to the time adjusted by user will be disturbed frequently by the notifications like in the figure until he stops the alarm by voice recognition. After this , as an input , this will be sent to the Beeminder so that the task is done.

Figure 22: Schedule Alarm Notification

5.6. Interaction Viewpoint

5.6.1. Add Schedule Component

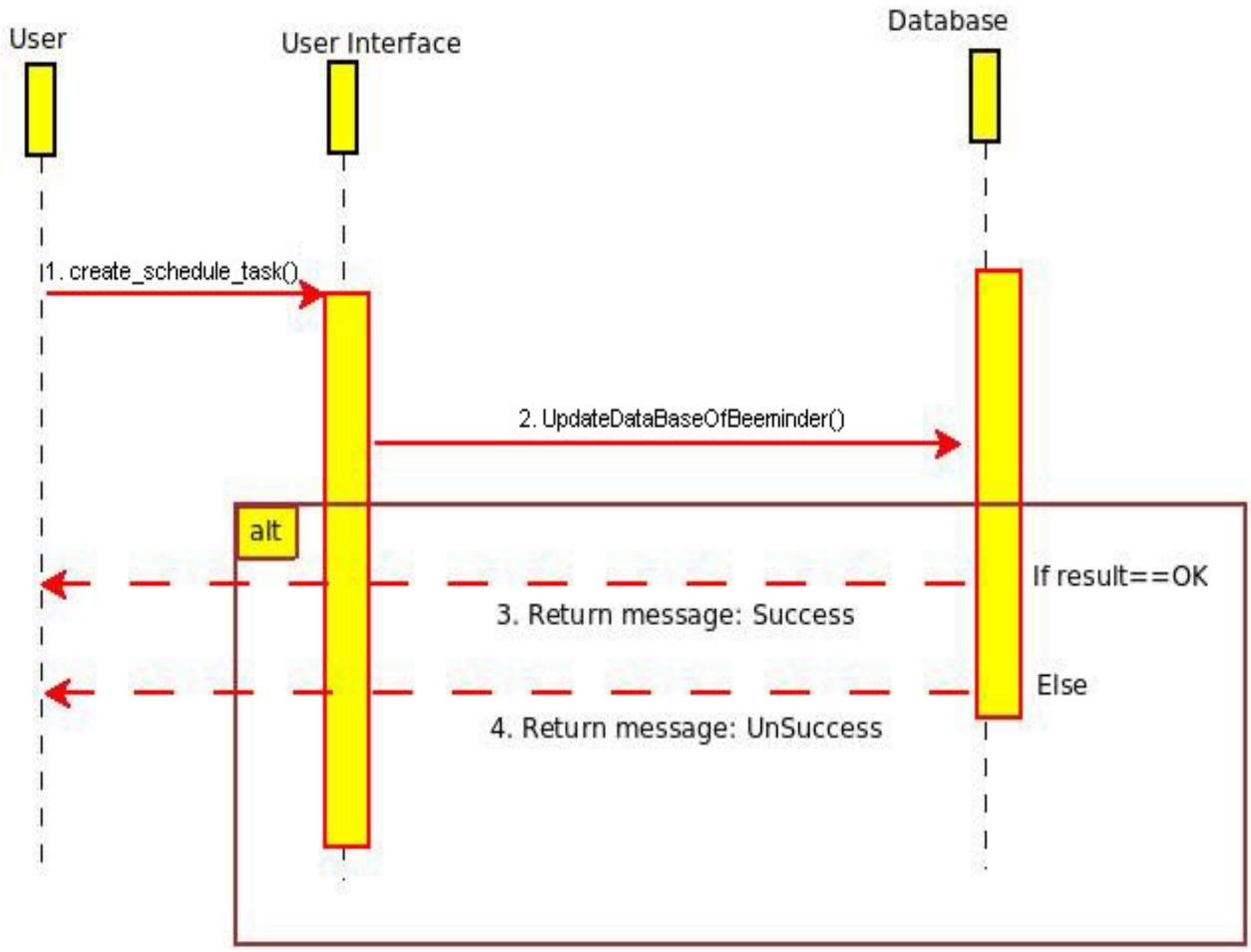

Figure 23 Sequence Diagram of Add Schedule Component

In application Schedule Alarm, the create_schedule_task() method will be used. The importance of this method is explained in the above and in SRS documentation. User can edit his/her schedule for getting alert according to it. The method UpdateDataBaseOfBeeminder() will add edited schedule by user to database of application and if result is okey the success message will return otherwise schedule is wanted to enter again.

5.6.2. Add Task Component

Over the six sub applications the same method will be used. This method will be used to open new task with the help of Beeminder account. Firstly, UpdateDataBaseOfBeeminder() method will be used to connect Beeminder server and send wanted new task request to it. According to gotten message back from Beeminder Server the application will redirect user to application main layout otherwise will return error message to user.

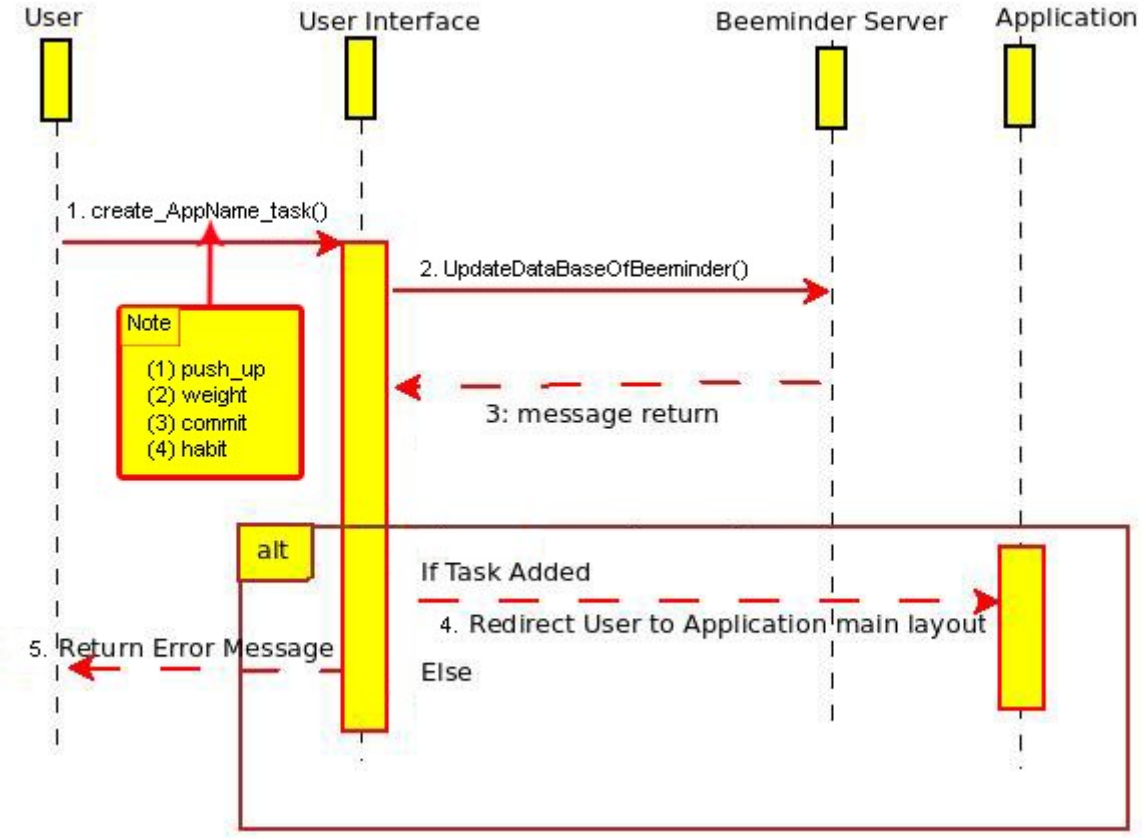

Figure 24 Sequence diagram for Add Task Component

5.6.3. Delete Task Component

Over the six sub applications the same method will be used. This method will be used to delete selected task with the help of Beeminder account. Firstly, UpdateDataBaseOfBeeminder() method will be used to connect Beeminder server and send wanted delete task request to it. According to response message from Beeminder Server the application will redirect user to application main layout otherwise will return error message to user. Application main layout will contain the task created by user.

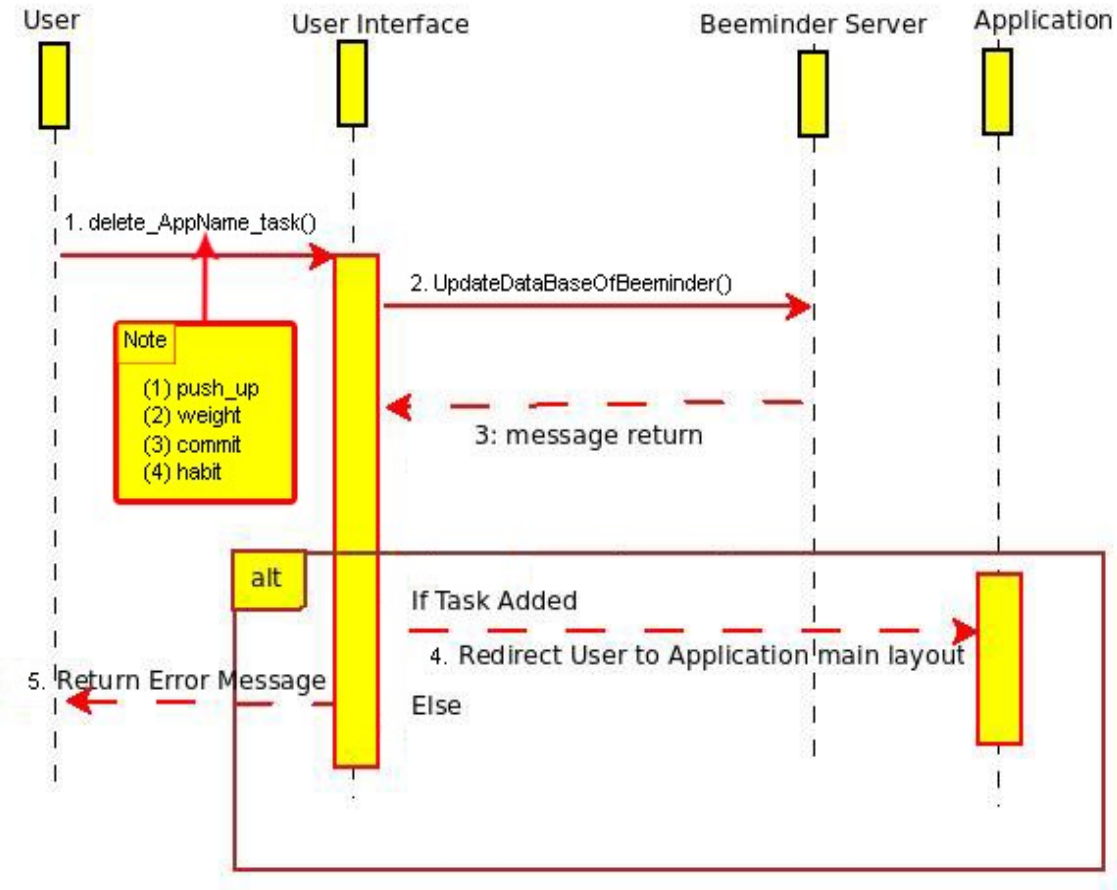

Figure 25 Sequence diagram for Delete Task Component

5.6.4. Edit Task Component

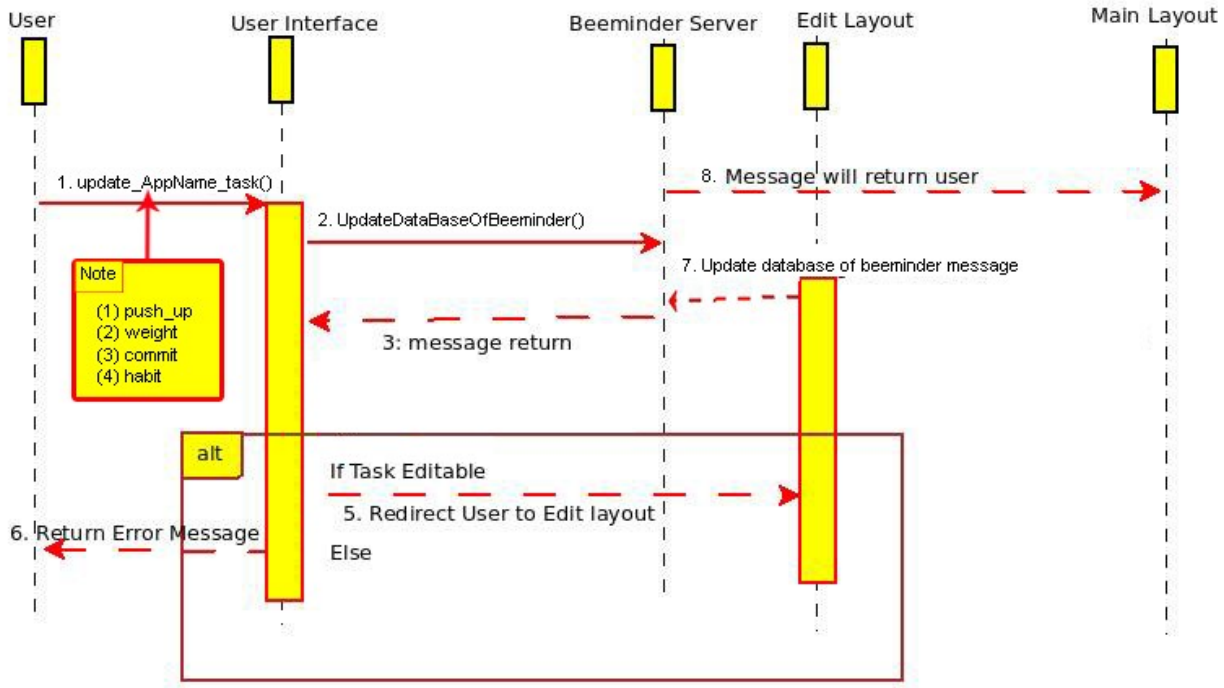

Figure 26 Sequence diagram for Edit Task Component

Over the six sub applications the same method will be used. This method will be used to edit selected and exist task with the help of Beeminder account. Firstly, UpdateDataBaseOfBeeminder() method will be used to connect Beeminder server and send wanted Edit/Update task request to it. According to response message from Beeminder Server the application will redirect user to application main layout otherwise will return error message to user. Application main layout will contain the task changed by user.

5.6.5. Photo Recognizer Component

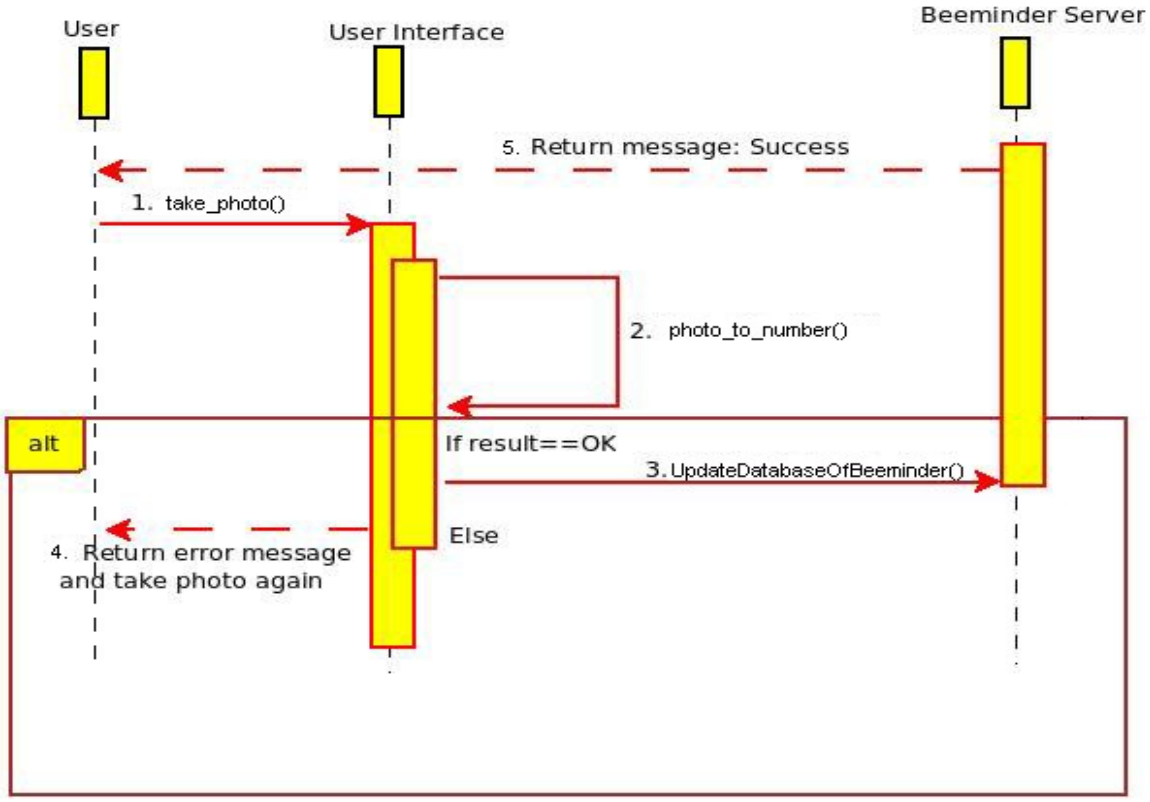

Figure 27 Sequence Diagram for Photo Recognizer Component

This seen component will be used in Weight-Loss Control application. User will start new request with take_photo() method. This method will take photo and send it to photo_to_number() method to recognize photo and convert given photo to a number. If response message is equal to OK , the application will send new request to Beeminder server for entering new data entry. Also application will save the photo in Database to ensure correctness of data for next try. And if the response from Beeminder server is okey, the sequence will be finished successfully. Otherwise, error message will return and wait for user to take photo again.