Vordiplom

Christian B. Sax;f.ochsn3r@gmail.com

PlexByte.com

30.11.2015 12:28

Mobile Communication App

***MoCap***



Table of Contents

[Management Summary 3](#_Toc436840284)

[What is MoCap 3](#_Toc436840285)

[System Context 3](#_Toc436840286)

[License addendum 4](#_Toc436840287)

[Additional Definitions 4](#_Toc436840288)

[Exception to Section 3 of the GNU GPL 4](#_Toc436840289)

[Conveying Modified Versions 4](#_Toc436840290)

[Object Code Incorporating Material from Library Header Files 5](#_Toc436840291)

[Combined Works 5](#_Toc436840292)

[Combined Libraries. 6](#_Toc436840293)

[Revised Versions of the GNU Lesser General Public License. 6](#_Toc436840294)

[Requirements Engineering 7](#_Toc436840295)

[Technological requirements 7](#_Toc436840296)

[Technology Matrix 7](#_Toc436840297)

[Solution Documentation 8](#_Toc436840298)

[Interaction Components 8](#_Toc436840299)

[Interfaces 8](#_Toc436840300)

[Chat 8](#_Toc436840301)

[Project 8](#_Toc436840302)

[Task 8](#_Toc436840303)

[Poll 9](#_Toc436840304)

[Reporting 9](#_Toc436840305)

[Accounting 9](#_Toc436840306)

[Backend Components 9](#_Toc436840307)

[Database Module 10](#_Toc436840308)

[WebService 10](#_Toc436840309)

[Functions 10](#_Toc436840310)

[Bindings 10](#_Toc436840311)

[Security Components 10](#_Toc436840312)

[Logging Components 10](#_Toc436840313)

[Technical Specification 11](#_Toc436840314)

[Interaction Components 11](#_Toc436840315)

[Interfaces 11](#_Toc436840316)

# Management Summary

## What is MoCap

MoCap is a windows form based application serving the purpose of managing small projects and simplifying communication. Besides typical chat functions, like self-destructing messages etc., the goal is to create an app that allows you to chat with people, setup simple projects, where you invite people to contribute to, distribute tasks among people who joined your project and track expenses of your endeavor.

Tasks created can be assigned manually, automatically or ever through a poll, where project members vote for the owner of the task.

To track costs, you can attach bills and invoices to a task. To avoid overspending, a certain budget can be set for each task to be accomplished.

Another focus of the app is security, thus any communication is encrypted before it leaves the device, ensuring that secured content only is being transmitted and persisted eventually.

Finally, the integration to the operating system allows alerts and notifications popping up in time anytime.

MoCap is the next generation of communication, helping people to stay connected and get projects going without a massive planning overhead. Try it now and see how MoCap will help you ease your day.

## How It Works

To get started you will need to register and open an account at our website. It won’t take long and we are not asking the hell out of you. All we need is a user name, a valid email address and you phone number. That’s it and you’re all set.

In case you want to share more info with users in your directory you can, at any time, by filling in your details in the admin section (see chapter [Admin](#_Admin)).

MoCap is a project driven model. Everything you do typically represent a project, may it be organizing a simple event with your friends or a complex design project that involves others people’s know how.

Besides the chat feature, which is of course available outside of a project too, any interaction is mapped to a project. Thus you start by creating a project.

We tried to keep the security concept simple and easy to follow. Hence there are three types of human actors only;

* the “Creator”, which actually has full control over any interaction assigned to his / her project
* the “Owner”, who, besides the “Creator”, in case they differ, has full control over any item in his queue
* and finally the “User”, who has read permissions only, unless an item / interaction is assigned to him, which then classifies him as “Owner”.

So once you created your project and assigned members to it, you can add / run / open

* **Tasks**Representing a piece of work, which can consist out of tasks being assigned to the task (called *Subtasks*). A task has several attributes, which are explained in more detail later on (see chapter [Interaction Components/Tasks](#_Task)).
* **Chats**Opening a chat on behalf of a project always includes all project members in the distribution list (see chapter [Interaction Components/Chat](#_Chat)).
* **Polls**You can create project polls, which allows project members to vote for an option and assign tasks based on the polls outcome(see chapter [Interaction Components/Poll](#_Poll)).
* **Accounting**You can enable accounting on your project, which allows you to set a budget on the project itself or individual tasks. Users contributing to a project can then add invoices and bills to it, reflecting the costs effectively accumulated (see chapter [Interaction Components/Accounting](#_Accounting)).
* **Reports**You can run reports against you projects, to stay informed and supervise progress as well as costs (see chapter [Interaction Components/Reporting](#_Reporting)).

After you have setup your project users can start participating and get the project going.

MoCap will also take care of a seamless and uninterrupted service by persisting any interaction within the system on a central database in the internet. No worries, your data is kept save and encrypted for security purposes (see chapter [Security Components](#_Security_Components)). Anything leaving or entering MoCap is encrypted, making it nearly impossible for a man in the middle attack.

## System Context



## License addendum

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007  
Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

### Additional Definitions

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

### Exception to Section 3 of the GNU GPL

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

### Conveying Modified Versions

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

1. under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

### Object Code Incorporating Material from Library Header Files

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the object code with a copy of the GNU GPL and this license document.

### Combined Works

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

d) Do one of the following:

1. Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

### Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

### Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

# Requirements Engineering

## Technology

1. The application must run on Windows 8 or higher
2. The technology used must offer GUI design through its IDE
3. Technology must offer OS integration to generate alarm and / or notifications
4. Allow MVC programming pattern
5. Technology must be user friendly, easy to handle and good documentation
6. Programming language must be versatile with regards to platform
7. Technology must offer great efficiency when it comes to GUI design

### Descision Matrix

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Descision Matrix** | | | | | | | |  |
|  | **Runs on Win8+** | **GUI Design** | **OS Integration** | **MVC Capability** | **User Friendly** | **Portability** | **Efficiency** |  |
| WinForm C# | 10 | 10 | 10 | 10 | 9 | 9 | 10 |  |
| ASP.net c# | 10 | 9 | 8 | 10 | 7 | 10 | 7 |  |
| Java JSF | 10 | 8 | 7 | 10 | 6 | 10 | 6 |  |
| Java Swing | 8 | 10 | 9 | 10 | 9 | 10 | 8 |  |
|  |  |  |  |  |  |  |  |  |
| **Criterion Weight** | 10 | 10 | 8 | 10 | 10 | 8 | 10 |  |
|  |  |  |  |  |  |  |  |  |
| **Weighted Scores** | | | | | | | |  |
|  | **Runs on Win8+** | **GUI Design** | **OS Integration** | **MVC Capability** | **User Friendly** | **Portability** | **Efficiency** | **Total Score** |
| WinForm C# | 100 | 100 | 80 | 100 | 90 | 72 | 100 | 642 |
| ASP.net c# | 100 | 90 | 64 | 100 | 70 | 80 | 70 | 574 |
| Java JSF | 100 | 80 | 56 | 100 | 60 | 80 | 60 | 536 |
| Java Swing | 80 | 100 | 72 | 100 | 90 | 80 | 80 | 602 |

## UI Design

## Interaction Components

The components listed here offer some kind of interaction between Users themselves or user(-s) to systems or vice versa. Interaction in this context means “non static”, either a user or the system has to do something and interact with someone.

You will find the textual requirements for each type of interaction, which will be further refined and brought into a technical specification in the “Technical Specification” section. Refer to the table of contents to navigate through this document

### Chat

A chat allows you to send and receive messages to and from other users, in private chats between two individual users or in group chats between multiple users. The chat can also be used to forward messages to other users.

The chat is the only component, which exists outside of a project, except the project chat.

#### Project creating Group chat

Once a new project is created it automatically creates a group chat with all the members from the project in it. It is not possible to invite or kick users from the chat. If new users join a project, they also join the project chat. If a member is kicked from a project, they also drop out from the chat.

#### Use Cases

Below all chat use cases are documented and visually lined out.

##### Diagram



##### -CreateChat

|  |  |
| --- | --- |
| **Use Case Id:** | 18 |
| **Use Case Name:** | Create |
| **Actors:** | User |
| **Description:** | The user creates a new chat |
| **Preconditions:** | 1. User is logged in 2. User opened the chat panel |
| **Normal Flow:** | 1. User clicks "New Chat" button 2. User enters ID of another user 3. User clicks "Create Chat" button |
| **Alternative Flow:** | 3a. The user cancels the create request  4. Dialog closes |
| **Exceptions:** | None |

##### -Creategroupchat

|  |  |
| --- | --- |
| **Use Case Id:** | 19 |
| **Use Case Name:** | Create Group Chat |
| **Actors:** | User |
| **Description:** | The user creates a new chat |
| **Preconditions:** | 1. User is logged in 2. User opened the chat panel |
| **Normal Flow:** | 1. User clicks "New Group Chat" button 2. User enters a title 3. User enters a description of the chat 4. User invites members to chat 5. User clicks "Save" button |
| **Alternative Flow:** | 5a. The user cancels the create group request  6. Dialog closes |
| **Exceptions:** | None |

##### -Invite

|  |  |
| --- | --- |
| **Use Case Id:** | 20 |
| **Use Case Name:** | Invite |
| **Actors:** | User |
| **Description:** | The user invites other users to an existing group chat |
| **Preconditions:** | 1. User is logged in 2. User has created a group chat |
| **Normal Flow:** | 1. User enters the group chat 2. User clicks "Options" button 3. User clicks "Invite User" button 4. User enters ID of another user 5. User clicks "Invite" button  6. Invitation sent |
| **Alternative Flow:** | 5a. The user cancels the invite request  6. Dialog closes |
| **Exceptions:** | None |

##### -SendMessages

|  |  |
| --- | --- |
| **Use Case Id:** | 21 |
| **Use Case Name:** | SendMessage |
| **Actors:** | User |
| **Description:** | The user sends a message in a chat |
| **Preconditions:** | 1. User is logged in 2. User has created a chat |
| **Normal Flow:** | 1. User enters a chat panel 2. User enters a message 3. User clicks "Send" button |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

##### -ReceiveMessage

|  |  |
| --- | --- |
| **Use Case Id:** | 22 |
| **Use Case Name:** | ReceiveMessage |
| **Actors:** | User |
| **Description:** | The user receives a message in a chat from an other user |
| **Preconditions:** | 1. User is logged in 2. User is in a chat |
| **Normal Flow:** | 1. User automaticly receives messages other users have sent |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

##### -ForwardMessage

|  |  |
| --- | --- |
| **Use Case Id:** | 23 |
| **Use Case Name:** | ForwardMessage |
| **Actors:** | User |
| **Description:** | The user forwards a message sent by an other user |
| **Preconditions:** | 1. User is logged in 2. User received message from an other user |
| **Normal Flow:** | 1. User enters chat with received message 2. User clicks "Option" button 3. User clicks "Forward Message" button 4. User marks message(s) 5. User specifies chat in wich to send message 6. User clicks "Send" button  7. Message sent |
| **Alternative Flow:** | 6a. The user cancels the forward request  7. Dialog closes |
| **Exceptions:** | None |

##### -Delete

|  |  |
| --- | --- |
| **Use Case Id:** | 24 |
| **Use Case Name:** | Delete |
| **Actors:** | User |
| **Description:** | The user deletes a chat |
| **Preconditions:** | 1. User is logged in 2. User opened the chat panel |
| **Normal Flow:** | 1. User clicks "Option" button 2. User clicks "Delete Chat" button 3. User selects the chat to be deleted 4. User clicks “Delete” button 5. Process ends |
| **Alternative Flow:** | 4a. The user cancels the delete request  5. Dialog closes |
| **Exceptions:** | None |

##### -LeaveGroupChat

|  |  |
| --- | --- |
| **Use Case Id:** | 25 |
| **Use Case Name:** | LeaveGroupChat |
| **Actors:** | User |
| **Description:** | The user leaves a group chat he is member of |
| **Preconditions:** | 1. User is logged in  2. User opened the chat panel 3. User has entered a group chat |
| **Normal Flow:** | 1. User enters the group chat 2. User clicks "Options" button 3. User clicks "Leave Group Chat" button  4. User clicks "Leave" button  5. Invitation sent |
| **Alternative Flow:** | 4a. The user cancels the leave request  5. Dialog closes |
| **Exceptions:** | None |

### Project

A project combines all components except the chat. It is a closed system that contains tasks, polls, balance and a group chat for all members of a project. Only users in a project can be assign to tasks or invited to polls. Every user can belong to more than one project.

#### Use Cases

Below all project use cases are documented and visually lined out.

##### Diagram



##### -Create

|  |  |
| --- | --- |
| **Use Case Id:** | 26 |
| **Use Case Name:** | Create |
| **Actors:** | User |
| **Description:** | The user creates a new project |
| **Preconditions:** | 1. User is logged in 2. User opened the project panel |
| **Normal Flow:** | 1. User clicks "New Project" button  2. User enters a title 3. User enters a description of the project 4. User specified the due date (if any)  5. User specifies if poll or balance is included in project 6. User enters ID of another user(s) 7. User clicks "Create Project" button  8. Project created |
| **Alternative Flow:** | 7a. The user cancels the create request  8. Dialog closes |
| **Exceptions:** | None |

##### -Invite

|  |  |
| --- | --- |
| **Use Case Id:** | 27 |
| **Use Case Name:** | Invite |
| **Actors:** | User |
| **Description:** | The user invites other users to an existing project |
| **Preconditions:** | 1. User is logged in  2. User opened the project panel 3. User has created a project |
| **Normal Flow:** | 1. User enters the project  2. User clicks "Options" button 3. User clicks "Invite User" button 4. User enters ID of another user 5. User clicks "Invite" button  6. Invitation sent |
| **Alternative Flow:** | 4a. The user cancels the invite request  5. Dialog closes |
| **Exceptions:** | None |

##### -Delete

|  |  |
| --- | --- |
| **Use Case Id:** | 28 |
| **Use Case Name:** | Delete |
| **Actors:** | User |
| **Description:** | The user deletes a project |
| **Preconditions:** | 1. User is logged in 2. User opened the project panel |
| **Normal Flow:** | 1. User enters the project  2. User clicks "Options" button  3. User clicks “Delete Project” button 4. User clicks the “Delete” button 5. Process ends |
| **Alternative Flow:** | 4a. The user cancels the delete request  5. Dialog closes |
| **Exceptions:** | None |

##### -LeaveProject

|  |  |
| --- | --- |
| **Use Case Id:** | 29 |
| **Use Case Name:** | LeaveProject |
| **Actors:** | User |
| **Description:** | The user leaves a project |
| **Preconditions:** | 1. User is logged in 2. User opened the project panel |
| **Normal Flow:** | 1. User enters the project  2. User clicks "Options" button 3. User clicks "Leave Project" button 4. User clicks the “Leave Project” button  5. User clicks the “Delete Project” button 6. Process ends |
| **Alternative Flow:** | 4a. The user cancels the leave request  5. Dialog closes  5a. The User cancels the delete request  6. Dialog closes |
| **Exceptions:** | None |

##### -KickUser

|  |  |
| --- | --- |
| **Use Case Id:** | 30 |
| **Use Case Name:** | LeaveProject |
| **Actors:** | ProjectCreator |
| **Description:** | The creator of a project kicks a user from it. |
| **Preconditions:** | 1. Creator is logged in 2. Creator opened the project panel |
| **Normal Flow:** | 1. Creator enters the project  2. Creator clicks "Options" button 3. Creator clicks "Kick User" button 4. Creator clicks un user name  5. Creator clicks the “Kick User” button 6. Process ends |
| **Alternative Flow:** | 5a. The user cancels the kick user request  6. Dialog closes |
| **Exceptions:** | None |

### Task

A task represents a piece of work one has to accomplish, typically until a date specified. It can consist out of many other subtasks which, as a whole, represent specific work.

As an Example:

The task “Go Shopping” consists, of subtasks  
1. “Get milk, honey and wheat”   
2. “Fill-up Gas”

In case a task contains subtasks, they automatically belong to the same project. You can enrich the task with a description and estimate its duration and cost. Additionally, you can specify alarms to be created if the due date is in danger. Once the alarm is raised, the owner can reassign the tasks to someone else or react on it by initiating a chat.

#### Manually assigning a task

Once a task is created and assigned to a project you can delegate it to any member of the project group. Manually assigned tasks do not consider the hours a person has dedicated to the project. Hence you need to make sure the task is assigned to a person dedicating enough time to this project or have the person to dedicate more time.

#### Poll assigning a task

You can assign a task through the result of a poll you created. To do that you create a project poll, through which members vote for the person to complete the task. Remember, that this method of assigning a task does not consider the hours, the person that got assigned to the task, has dedicated to this project.

#### Auto assigning a task

When selecting to auto assign the task, the due date and a priority define which person gets assigned to the task. The priority is used to determine the importance compared to other tasks this person may has been assigned to. The system then tries to find a project member that contributes enough time towards this project and has enough capacity to complete the task on time. In other words, if a task will take 5 hours to complete and has to be completed within 2 days, a project member contributing 1 hour a day will not be assigned to this task.

When completing the task, the person is requested to enter the time and money (if at any) spent on the task. The project owner is then notified of the completion and the project balance is updated accordingly.

#### Class Diagram

#### Sequence Diagram



#### Use Cases

Below all task use cases are documented and visually lined out.

##### Diagram



##### - Create

|  |  |
| --- | --- |
| **Use Case Id:** | 1 |
| **Use Case Name:** | Create |
| **Actors:** | User |
| **Description:** | The user creates a new task |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User clicks "New Task" button 2. User enters a title 3. User enters a description of the task 4. User estimates the duration 5. User specifies the start date 6. User specified the due date (if any) 7. User associates task with project 8. User specifies a budget (if any) 9. User selects either "auto-assign", "manual-assign" or "poll-assign" 10. User clicks "Save" button |
| **Alternative Flow:** | 9a. If the user selects "poll-assign"  10. => Goto Use Case **"Create" (poll)** 7b. Project does not exist yet  7. => Goto Use Case **"Create" (project)** |
| **Exceptions:** | None |

##### - Update

|  |  |
| --- | --- |
| **Use Case Id:** | 2 |
| **Use Case Name:** | Update |
| **Actors:** | User |
| **Description:** | The user updates task information |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User selects the task to be updated from within the task list 2. User updates corresponding fields 3. User saves changes |
| **Alternative Flow:** | 2a. If the user adds subtasks to the task  Goto Use Case **"Create" (task)** 2b. If the user re-assigns the task  Goto Use Case **"Assign" (task)** |
| **Exceptions:** | None |

##### - Delete

|  |  |
| --- | --- |
| **Use Case Id:** | 3 |
| **Use Case Name:** | Delete |
| **Actors:** | User |
| **Description:** | The user deletes a task |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User selects the task (-s) to be deleted 2. User clicks the "delete task" button 3. User confirms the deletion |
| **Alternative Flow:** | 3a. If the user cancels the deletion  3. => abort process |
| **Exceptions:** | None |

##### - Forward

|  |  |
| --- | --- |
| **Use Case Id:** | 4 |
| **Use Case Name:** | Forward |
| **Actors:** | User |
| **Description:** | The user forwards the task to another user |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User selects the task (-s) to be forwarded 2. User clicks the dispatch button 3. User selects either "auto-dispatch", "manual-dispatch" or "poll-dispatch" 4. User clicks "save" button |
| **Alternative Flow:** |  |
| **Exceptions:** | 2a. The user is not permitted to forward the task  3. => abort process |

##### - UpdateProgress

|  |  |
| --- | --- |
| **Use Case Id:** | 5 |
| **Use Case Name:** | UpdateProgress |
| **Actors:** | User |
| **Description:** | The user updates the progress of the task |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User selects the task (-s) to be updated 2. User changes the progress value 3. User clicks the "save" button 4. System updates task progress and subtask(-s) progress accordingly 5. Dialog closes |
| **Alternative Flow:** | 2a. The user updates a subtask  3. => Goto "UpdateProgress" use case for subtask  4. System updates the progress based on subtask status  5. User clicks the "save" button  6. Dialog closes 2b. User adds a subtask to the task  3. => Goto use case "Create" (task)   4. => Continue as per 2a.4 2c. User deletes a subtask  3. => Goto use case "Delete" (task)  4. => Continue as per 2a.4 2d. User completes the task  3. => Goto use case "Complete" (task) 2e. User cancels the dialog  3. Dialog closes |
| **Exceptions:** | 2a. The user specifies a negative or lower value than before  3. Message is shown  4. => Goto 2. |

##### - ManualDispatch

|  |  |
| --- | --- |
| **Use Case Id:** | 6 |
| **Use Case Name:** | ManualDispatch |
| **Actors:** | User |
| **Description:** | The user dispatches the task manually |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User selects the task (-s) to be dispatched 2. User selects "Manual Dispatch" 3. User selects project member to assign the task 4. System evaluates availability of user => Use Case "GetIdealMembers" (task) 5. System displays availability stats 6. User clicks the "Dispatch" button 7. System dispatches the task to selected member 8. Dialog closes |
| **Alternative Flow:** | 5a. The selected user does not have anough availability  6. => Goto use case "NotifyInsufficientAvailability" (task)  7. Dialog closes 2a. User selects "Auto-Dispatch"  3. System evaluates availability of user => Use Case "GetIdealMemebers" (task)  4. User clicks "Dispatch" button  5. Dialog closes   6. System selects project member  7. System assigns task to member 2b. User selects "Poll-Dispatch"  3. => Goto use case "Create" (Poll)  4. Dialog closes  5. System assigns task based on poll result |
| **Exceptions:** | none |

##### - AutoDispatch

|  |  |
| --- | --- |
| **Use Case Id:** | 7 |
| **Use Case Name:** | AutoDispatch |
| **Actors:** | System |
| **Description:** | The system dispatches a task based on availability, and preferences defined |
| **Preconditions:** | 1. A task exists or was created 2. Auto-Dispatch function was called |
| **Normal Flow:** | 1. System retrieves project members 2. System calculates scor card by executing "Get Ideal Members" use case 3. System identifies member with highest score card 4. System assigns task to user 5. System raises "Task Assigned" event 6. System adds users to task |
| **Alternative Flow:** | 2a. No due date specified  3. Randomly select user  4. => Goto use case step 4 2b. System does not find user with sufficient availability  3. => Raise "Insufficient Resources" Exception  4. End process |
| **Exceptions:** | 2b. Insufficient availability |

##### - Complete

|  |  |
| --- | --- |
| **Use Case Id:** | 8 |
| **Use Case Name:** | Complete |
| **Actors:** | System |
| **Description:** | The system detects that the task is completed after the user has updated the task progress |
| **Preconditions:** | 1. User has updated the task progress 2. Task is completed after update progress ends |
| **Normal Flow:** | 1. System marks the task as "Completed" 2. System notifies the users (Creator and current task owner) 3. System checks if the task is a subtask and updates the parent task if required 4. System raises "Task Completed" event |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

##### - GetMatchingMembers

|  |  |
| --- | --- |
| **Use Case Id:** | 9 |
| **Use Case Name:** | GetMatchingMembers |
| **Actors:** | System |
| **Description:** | The system identifies members able to complete the task until due date whithout over calculating hours dedicated to this project |
| **Preconditions:** | 1. A task exists or was created 2. Task was selected in Task panel 3. Task opened 4. Due Date was specified 5. Task duration was specified |
| **Normal Flow:** | 1.System calcutates score cards for each member 2. System identifies member offering sufficient time to this project 3. Process ends |
| **Alternative Flow:** | 1a. No due date specified  2. Display all users  4. Process ends 2b. System does not find user with sufficient availability  3. => Raise "Insufficient Resources" Exception  4. End process |
| **Exceptions:** | 2b. Insufficient availability |

#### GUI Mockup

##### Overview



##### Detail View



### Poll

A poll allows users in your project to vote for one or multiple pre-defined option(-s), based on your configuration. In some scenarios it might be useful to allow user specifying an individual option, besides the ones pre-defined. The vote can be limited to a specific time / date range and the users eligible can be specified individually.

Once the poll has ended, which happens either after the date / time range specified elapsed or when every eligible user has voted, the results are being sent to either all, a specific group or the creator only.

If the poll was to vote for a task owner, both the current task owner as well as the future owner are notified together with the users specified and the owner of the poll. The task is then taken off the current user and assigned to the new user.

#### Use Cases

Below all task use cases are documented and visually lined out.

##### Diagram



##### - Create

|  |  |
| --- | --- |
| **Use Case Id:** | 10 |
| **Use Case Name:** | Create |
| **Actors:** | User |
| **Description:** | The user is creating a new Poll |
| **Preconditions:** | 1. User is logged in 2. User opened the poll panel |
| **Normal Flow:** | 1. User clicks the create poll button 2. User specifies a poll title 3. User enters a description of the poll (optional) 4. User defines the date this poll ends 5. User defines whether or not custom options are allowed to vote for 6. User enters a list of options to vote 7. User selects the people eligible to vote 8. User specifies the number of votes per users 9. User saves the vote 2. System identifies member offering sufficient time to this project 3. Process ends |
| **Alternative Flow:** | 4a. No end date specified  4. Poll ends once each user has voted  5. Goto use case step 5 5a. User selects a task to distribute  6. Options are set automatically (including all project members)  7. Eligible users are project members only  8. Goto use case step 8 |
| **Exceptions:** | None |

##### - CreateOptions

|  |  |
| --- | --- |
| **Use Case Id:** | 11 |
| **Use Case Name:** | CreateOptions |
| **Actors:** | User |
| **Description:** | The user is creating a new option selectable in a poll |
| **Preconditions:** | 1. User is logged in 2. User opened the poll panel |
| **Normal Flow:** | 1. User clicks the create option button 2. User specifies an option title 3. User enters a description for the option (optional) 9. User saves the option |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

##### - Delete

|  |  |
| --- | --- |
| **Use Case Id:** | 12 |
| **Use Case Name:** | Delete |
| **Actors:** | User |
| **Description:** | The user deletes a poll |
| **Preconditions:** | 1. User is logged in 2. User opened the poll panel |
| **Normal Flow:** | 1. User selects the poll to be deleted 2. User clicks the delete button 3. Process ends |
| **Alternative Flow:** | 2a. The user cancels the delete request  3. Dialog closes |
| **Exceptions:** | None |

##### - Complete

|  |  |
| --- | --- |
| **Use Case Id:** | 13 |
| **Use Case Name:** | Complete |
| **Actors:** | User / System |
| **Description:** | Occurs once the poll is completed. They happens either when the due date expires or all users have left their vote |
| **Preconditions:** | 1. User is logged in 2. User opened the poll panel |
| **Normal Flow:** | 1. User votes 2. System checks if all users have voted or due date expired 3. System closes the poll 4. System notifies users 5. System marks poll as completed |
| **Alternative Flow:** | 2a. Some votes are still open  3. The process ends 5a. The poll was a assign poll  6. The system assigns the task to user  7. System notifies the task owner  8. The process ends |
| **Exceptions:** | None |

##### - Vote

|  |  |
| --- | --- |
| **Use Case Id:** | 14 |
| **Use Case Name:** | Vote |
| **Actors:** | User |
| **Description:** | The user votes for an option |
| **Preconditions:** | 1. User is logged in 2. User opened the poll |
| **Normal Flow:** | 1. User selects an option 2. User clicks save button to submit his vote 3. Dialog closes |
| **Alternative Flow:** | 1a. User files a custom option  2. The user defines a new option  3. User saves the new option  4. User selects the new option  5. User saves the vote  6. The dialog closes |
| **Exceptions:** | None |

##### - Invite

|  |  |
| --- | --- |
| **Use Case Id:** | 15 |
| **Use Case Name:** | Invite |
| **Actors:** | User |
| **Description:** | The user invites participants for a poll |
| **Preconditions:** | 1. User is logged in 2. User opened the poll |
| **Normal Flow:** | 1. User opens the poll 2. User clicks the participants button 3. User adds participants from either a project or individually from his / her addressbook 4. User saves the poll 5. The dialog closes 6. The system notifies the user 7. Process ends |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

##### - AssignTask

|  |  |
| --- | --- |
| **Use Case Id:** | 16 |
| **Use Case Name:** | AssignTask |
| **Actors:** | System |
| **Description:** | Occurs if a poll has completed and a task was attached to it |
| **Preconditions:** | 1. Poll completed 2. Task attached to poll |
| **Normal Flow:** | 1. System couts most user votes 2. System assigns user to task 3. System updates the task 4. System updates the project 5. Process ends |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

#### GUI Mockups

##### Overview



##### Detail View

### Reporting

### Accounting

#### Balance

The Balance is an overview of all the expenses from a project. It shows the expenses of every task in a project, compares the expenses from every member and gives an overview of the expenses from all tasks a user owns.

##### Use Cases

Below all balance use cases are documented and visually lined out.

###### Diagram



###### -AttachBalance

|  |  |
| --- | --- |
| **Use Case Id:** | 31 |
| **Use Case Name:** | AttachBalance |
| **Actors:** | User |
| **Description:** | The user creates a balance object and attaches it to a task |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel  3. The task is assigned to the user |
| **Normal Flow:** | 1. User clicks "Add Balance" button 2. User enters expenses of the task  3. User clicks "Add" button |
| **Alternative Flow:** | 3a. The user cancels the attach request  4. Dialog closes |
| **Exceptions:** | None |

###### -AttachBill

|  |  |
| --- | --- |
| **Use Case Id:** | 32 |
| **Use Case Name:** | AttachBill |
| **Actors:** | User |
| **Description:** | The user attaches an image if the bill to a task |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel  3. The task is assigned to the user |
| **Normal Flow:** | 1. User clicks "Add Bill" button 2. User attaches image of the bill  3. User clicks "Add" button |
| **Alternative Flow:** | 3a. The user cancels the attach request  4. Dialog closes |
| **Exceptions:** | None |

###### -EditBalance

|  |  |
| --- | --- |
| **Use Case Id:** | 33 |
| **Use Case Name:** | EditBalance |
| **Actors:** | User |
| **Description:** | The user edits a balance object attached to a task |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel  3. The task assigned to the user has a balance attached |
| **Normal Flow:** | 1. User clicks "Edit Balance" button 2. User enters expenses of the task  3. User clicks "Edit" button |
| **Alternative Flow:** | 3a. The user cancels the edit request  4. Dialog closes |
| **Exceptions:** | None |

###### -ViewProjectBalance

|  |  |
| --- | --- |
| **Use Case Id:** | 34 |
| **Use Case Name:** | ViewBalance |
| **Actors:** | User |
| **Description:** | An overview of the expenses from all users for a project |
| **Preconditions:** | 1. User is logged in |
| **Normal Flow:** | 1. User clicks on the balance panel |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

###### -ViewBalance

|  |  |
| --- | --- |
| **Use Case Id:** | 35 |
| **Use Case Name:** | ViewBalance |
| **Actors:** | User |
| **Description:** | An overview of the expenses from the user for a project |
| **Preconditions:** | 1. User is logged in |
| **Normal Flow:** | 1. User clicks on the balance panel |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

#### Bills

The Bill is used to attach an image of a bill to a task as proof of the expenses used on a task.

Additionally to the owner of a task the creator of it and the creator of the project can also edit the file.

## Admin

## Settings

The “Settings” is used to personalize MoCob.

##### Use Cases

Below all settings use cases are documented and visually lined out.

###### Diagram

## Backend Components

### Database Module

#### Programmability

##### Stored Procedures

##### Views

##### Triggers

## WebService

### Functions

### Bindings

## Security Components

## Logging Components

# Technical Specification

## Interaction Components

### Interfaces

#### IComponent (Interface)

#### Task (Class)

##### Class Diagram

