Concentration, Order and Memory (COM), Mobile- and Web-applications for Adults with Attention Deficit Hyperactivity Disorder (ADHD)

Project Group:
Hyperactivity

Group Members:
Yvonne Le
Mathias Lindblom
Kim Malmros
Jesper Norberg
Marcus Nordström
Erik Odenman
Jennie Olsson
Martin Pettersson
Lucas Wiener

Revised Architectural Design Document

Version 1.1
2013-04-26

Table of Contents

Document Change Record

Version 0.1,* 2013-01-28*: Created the document (Martin)
Version 0.2,* 2013-02-17*: Added System Context diagrams and component table templates (Marcus, Martin)
Version 0.3,* 2013-02-24*: Made research and edited component tables for database components (Marcus, Martin, Mathias)
Version 0.4,* 2013-02-27*: Completed system overview, added database table (Marcus)
Version 0.4,* 2013-02-27*: Added Meeting Minutes (Martin)
Version 0.4,* 2013-02-27*: Edited layout for component section (Martin)
Version 0.4,* 2013-02-27*: Added 4 classes the component section (Mathias)
Version 0.5,* 2013-02-28*: Wrote purpose, overview of the document and software scope. Also added some acronyms and references. (Martin)
Version 0.5,* 2013-02-28*: Completed Introduction Section. (Martin)
Version 0.5,* 2013-02-28*: Wrote system context information and definitions for external interfaces.
Version 0.5, 2013-02-28: Completed Section 4: System Context (Martin)
Version 0.5,* 2013-02-28*: Wrote paragraph in design composition. (Martin)
Version 0.5,* 2013-02-28*: Added the rest of the client classes components and some of the Logic Application Layer (Mathias)
Version 0.5,* 2013-02-28*: Completed section 7: User requirements vs Compatibility Traceability Matrix (Mathias, Erik)
Version 0.5,* 2013-02-28*: Added definitions, links, and a checklist (Jesper)
Version 0.5,* 2013-02-28*: Changed wordings, proofread (Jesper)
Version 0.6,* 2013-02-29*: Added database values to components (Mathias)
Version 1.0,* 2013-02-29*: Wrote and added abstract. (Martin)
Version 1.0,* 2013-02-29*: Edited layout, formatted and finalized (Martin)
Version 1.1, 2013-04-22: Cross references corrected, headings updated, images updated, proofreading and wording changed (Kim)
Version 1.1, 2013-04-26: Proofread, formatted and finalized. (Martin)

Abstract

This document summarizes work made in the Architectural Design Phase of group Hyperactivity's project in the course DD1365 Software Engineering. As stated in the Project Handbook which the group has been provided, the architectural phase concerns the construction of an architectural model that relates to software requirements presented in the User Requirements Document. The Architectural Design Document presents the design in a top-level down fashion, where the system starts to be presented in a more abstract way and ends with visualization using UML and declarations of all system components. The architectural model will consist of a three-tier system model, where the client is based on the Android platform and the server written in Java using an application logic layer alongside a MySQL database. The two interdependent parts will communicate through JSON-RPC solution.

The last parts presents a short study on feasibility and resource estimates, where all system components are prioritized with a priority model. Efforts for future project tasks are estimated, and a risk analysis is also presented along with a compatible schedule.

The final part presents a matrix that visualizes relationships between system components and software requirements stated in the User Requirement Phase.

1. Introduction

1.1. Purpose

The purpose of the Revised Architectural Design Document (RADD) is to give an up to date and specific overview of the system design as it has been realized since the original ADD which built upon the User Requirements Document (URD). In terms of the *Project Handbook*, the purpose of the architectural design phase is to construct an architectural model that meets software requirements outlined in the earlier phase. This document is meant to justify the group's choice of architectural design where the design itself is intended to maximize cohesion and minimize coupling in order to ease future software maintenance and upgrades.

1.2. Scope of the Software

The application is a specialized forum-journal solution for individuals diagnosed with Attention Deficit Hyperactivity Disorder (ADHD) for the Android platform. The objective is to enable said individuals to connect with similar people and to aid in day to day struggle.

The system architecture is mainly based on a three tier client-server infrastructure consisting of external interfaces, a client side using the Android platform, an application logic layer written in Java, and a storage system managed with the open source MySQL database.

1.3. Definitions, Acronyms and Abbreviations

ADD	Architectural Design Document (*unused:* Attention Deficit Disorder)
ADHD	Attention Deficit Hyperactivity Disorder
Cohesion	The interdependencies between the various modules
Coupling	How related functions within a module are
SDK	Software Development Kit
API	Application Programming Interface
GUI	Graphical User Interface
UML	Unified Modeling Language
URD	User Requirements Document
RUP	Rational Unified Process

1.4. References

Information on difference between platform versions of the Android operating system:
http://developer.android.com/about/dashboards/index.html
Multitier Architecture:
http://www.linuxjournal.com/article/3508
Project Handbook:
http://www.nada.kth.se/~karlm/mvk/Project_Handbook.pdf
MySQL Connector, J Library:
http://dev.mysql.com/doc/refman/5.5/en/connector-j.html
JSON RPC-tutorial:
http://docs.webob.org/en/latest/jsonrpc-example.html

Facebook Authentication Technology:
https://developers.facebook.com/docs/concepts/login/
Waterfall model
R.Mall. Fundamentals of Software Engineering. Third edition. New Delhi: PHI Learning Private Limited, 2009. (pages 33-45)

1.5. Overview of the Document

The first part of the document gives a general overview of what is included. The main intention of the layout is to give the reader an understanding of the system and its components, first parts providing information in a very general way with lower complexity. The latter parts will focus on individual components of the architecture with less abstraction.

More specifically, the second part gives an overview of the entire system and its design. The third part provides context, listing all external interfaces with corresponding definitions. The fourth part provides an overview of design techniques used and also describes non-standard UML methods we have used.

After the system has been given a general overview an extensive fifth part will give detailed component information. This part will describe Java classes used and is not intended for the reader to dive into, but rather be seen as a declarative instruction of the UML diagrams which describes the architectural design more intuitively and visually.

The sixth part will summarise conclusions of a feasibility study around the architectural model. All components will be prioritised, ranging from economy to deluxe. Here we will also describe future project tasks and provide information on how the rest of the project will be allocated among project staff members. In the final part system components will be traced back to the requirements described in the *Revised User Requirements Document*.

2. System Overview

The system follows the classical three-tiered architecture model including database, business logic and graphical user interface. The database will be a MySQL database with the purpose of keeping track of all our forum data, users, user settings, threads etc.

The business logic tier acts as the communication medium between the client tier and the functional processing. All code that can be put on the business logic side will be put there primarily since keeping everything on the server side will make version handling a lot easier. Another benefit is that the client program will become more lightweight. One can argue that keeping too much logic on the server side will make the system sensitive to bad internet connection. However, as the target group lives in Sweden where mobile internet is relatively stable and as the main purpose of the program is to communicate over the internet, we think our choice is justified. Both the business logic and the Android GUI will be entirely written in Java.

The business logic and MySQL will coexist on the same physical server machine and will communicate with the MySQL Connector/J library. The client side and business logic will both be written in Java and will communicate through the JSON-RPC technology. This will make classes on the business-logic side appear as classes on the client side, even though their functions will run remotely. Since the standard does not enforce cryptography by default and our data in some cases needs to be secured from “sniffing” and “man-in-the-middle” attacks, we intend to initiate our connection over SSL. Also, because our target audience is all Android and mobile users, they will most likely be connected to *Facebook*. Because of the ease to enable authentication through Facebook’s API, we have decided to enable all authentication mechanisms through this.

Synchronization problems will naturally arise in some parts of the application when for instance some user A is in a thread that at the same time is being deleted by user B. In order to avoid unnecessary complexity in the software we intend to rely on a simple refresh button and refresh timers, that is, no asynchronous updating.

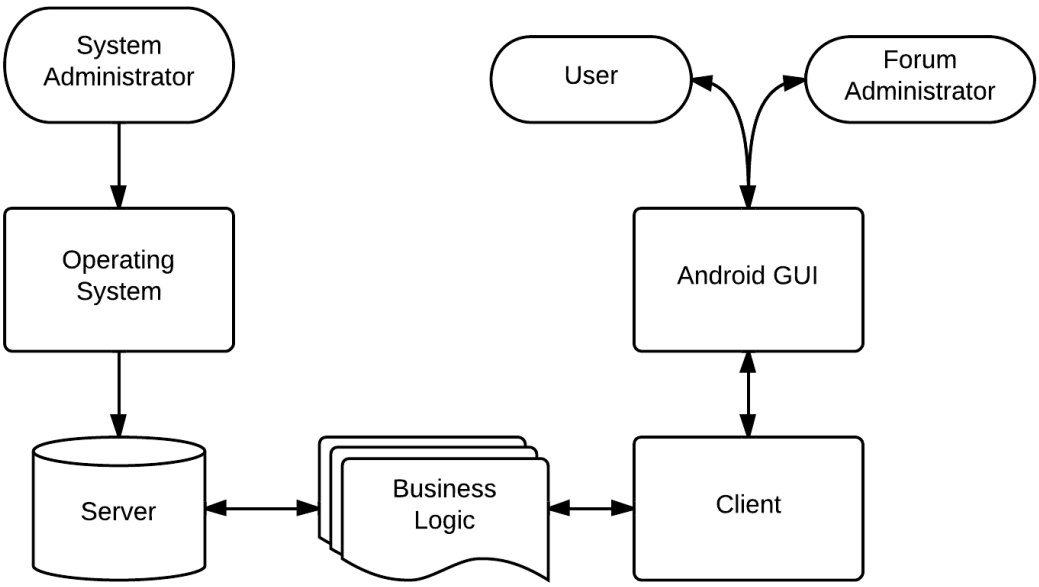
3. System Context

As previously mentioned, our system model is mainly based on the Three-Tier Database Architecture. This design depends on three interdependent components: an interface layer, an application logic layer and a storage layer.

Generally, the interface layer is supposed to include objects dealing with the end users, which in our case are the sole external interfaces; here, objects are mostly limited to graphical user interface elements such as interactive windows, forms, panels, web pages or similar. In our case, the objects in this layer are mostly limited to Activities and Fragments, which both are part of the Android interface platform.

Already described in the overview, we consider discovering the possibilities of using an interface between the client and the server based on the JSON-RPC solution, but this should be considered as an internal interface.

Both the logic and storage layer will be kept on an external server; objects here include Java classes intended for communication with the MySQL database and the storage itself. This system object has only one external interface, for the server administrator.



Depicting the System Context

3.1 External Interface Definitions

Not applicable.

4. System Design

4.1. Design Method

We are using the waterfall model in our project. The waterfall model is a sequential design process which assumes that a phase must be complete and correct before the following one begins. Therefore there should be no need to go back and make changes to the work in the earlier phases. However this is unfeasible in practice and one often has to go back and correct errors, but the goal is to do everything perfectly from the start. The progress is demonstrated by a steady flow downwards through the phases - requirements, design, implementation, verification and maintenance.

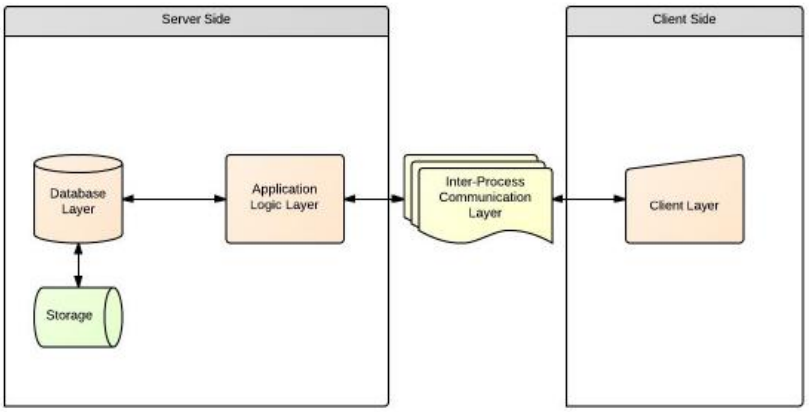
As such the requirement phase was already completed when we started with the system design. The system design is created with the software requirement as a base. Since we already had the requirements our next step was to go through and decide the structure. At first, we sketched our database. Furthermore, we discussed the application logic layer and compared different possible solutions. Finally the decision about the logic and navigation of the client was made. This work was similar to the Rational Unified Process, RUP.

4.2. Decomposition Description

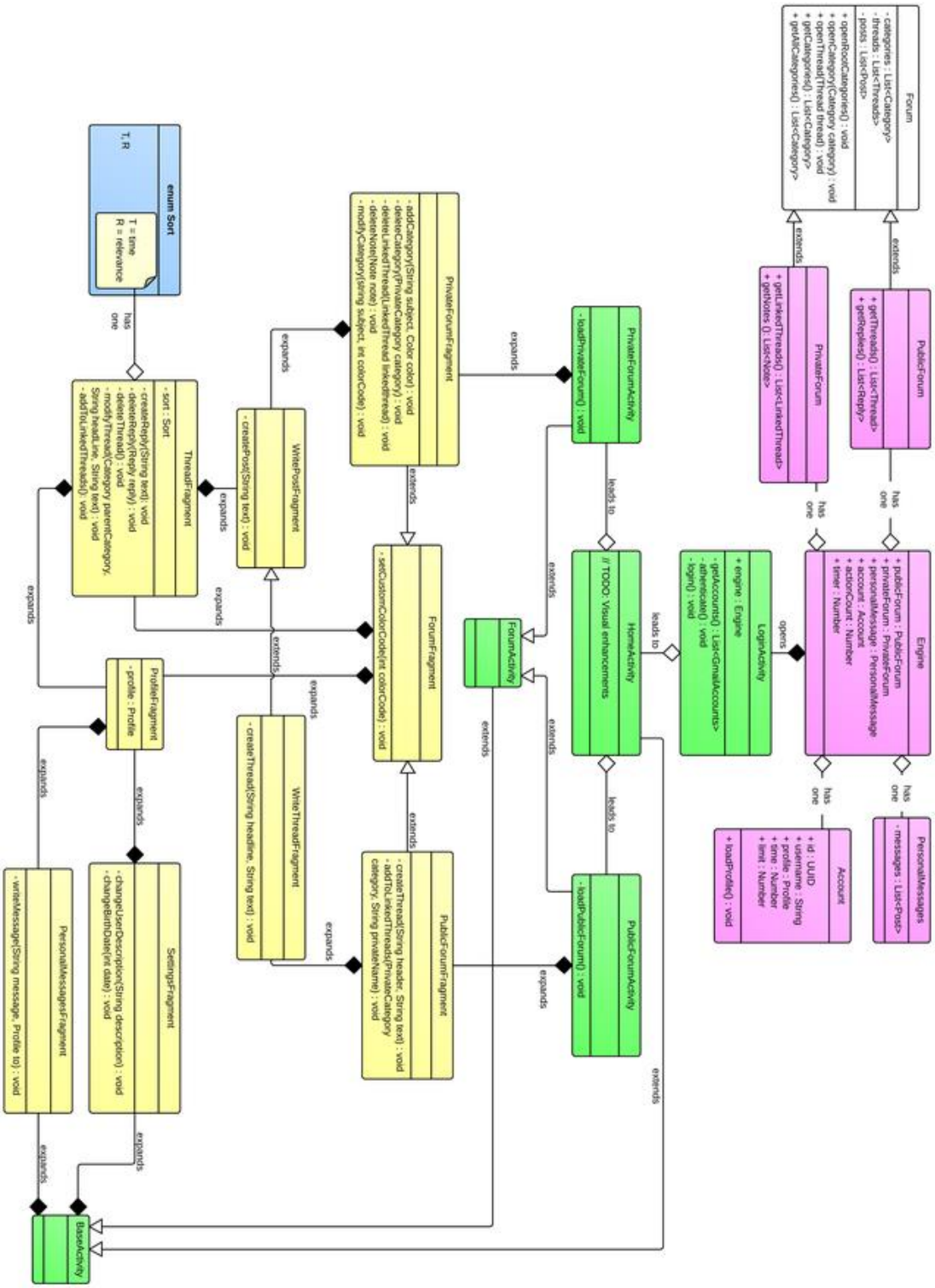
Shown below is a diagram and a top level view of the system's design. The illustration shows the major components which will be described in a more detailed manner in section 5: Component Description. Following sections provides two extensive illustrations with more complexity of the Client Orientation Logic, as well as the Business Logic part on the server side. Both these diagrams are made with standard conventions using Unified Modeling Language (UML) in order to be as detailed as possible in terms of component visualization.

Both these UML diagrams should be examined carefully by the reader who wants a detailed overview of the system and should be considered an essential tool of reference when examining all specifications of the components. These diagrams cover both server and client Java class files.

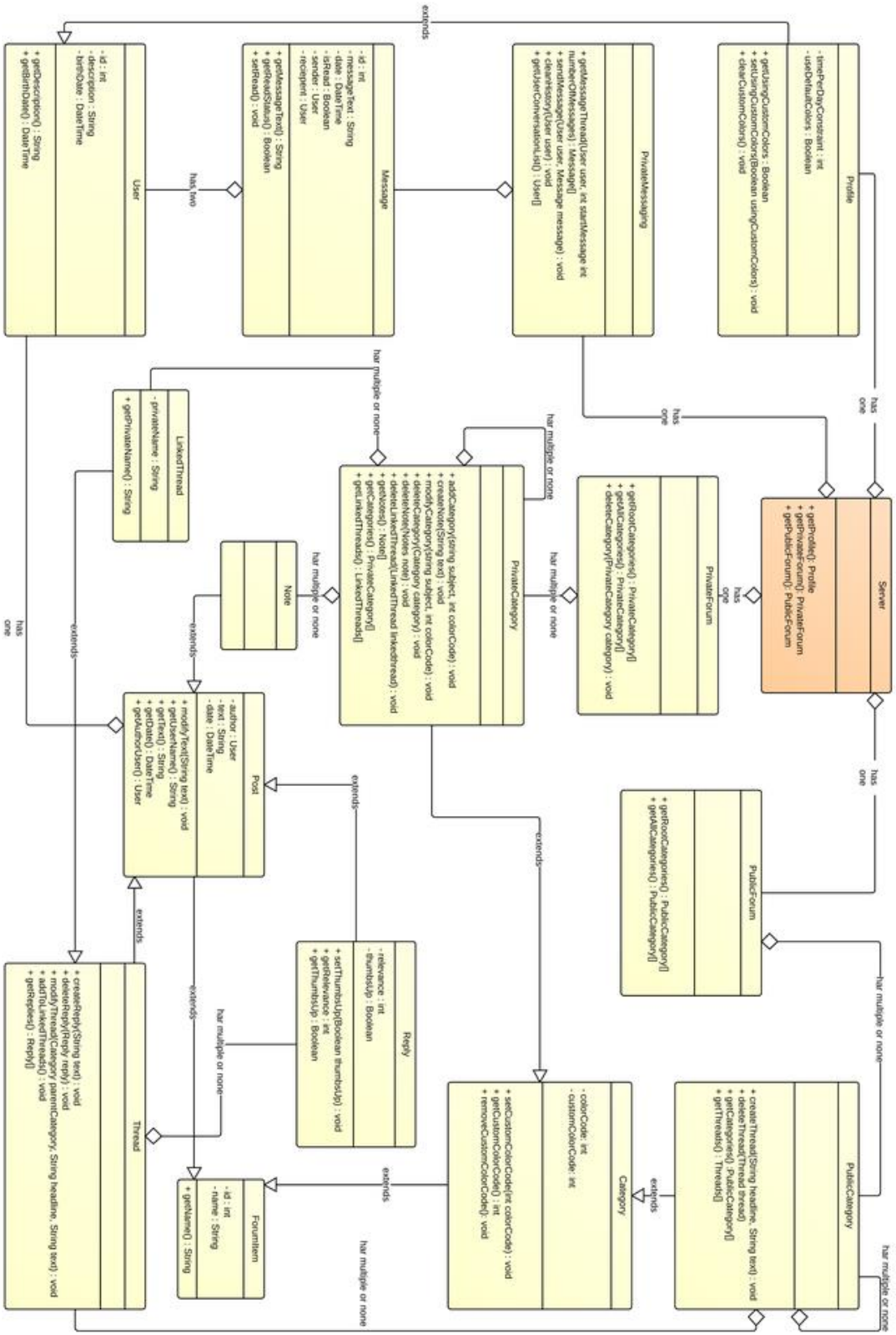
The only deviation from conventional UML is that we have decided to use is the term *expands* on the client side. When a class *expands* another it points out and illustrates the Android *Fragment* navigation flow of the client. For simplicity an Android fragment can be seen as HTML web pages; an expansion of fragments would in this analogy be considered as a corresponding hyperlink.



4.2.1 UML Diagram of Client Side: Android Platform



4.2.1 UML Diagram of Server Side: Business Logic



5. Component Description

Android programming concepts

Two concepts are of special importance when programming for Android; *activities* and *fragments*. An *activity* is a single, focused pane that a user can do such as viewing a map or playing a game. It has its own window that can take input and gets its own window to draw in and it can contain one or multiple *fragments*. A *fragment* is a modular part of an *activity* and cannot run on its own. The *fragment* can be a part of a GUI such as a button or a pane and will be destroyed along with its mother *activity*.

5.1 Database Components

5.1.1 PublicCategory

5.1.1.1 Type
Database table.

5.1.1.2 Purpose
Provide Category data to the business logic layer.

5.1.1.3 Function
Provide Category data to the business logic layer through standard MySQL queries.

5.1.1.4 Subordinates
None.

5.1.1.5 Dependencies
None.

5.1.1.6 Interfaces
The dataflow is controlled by MySQL.

5.1.1.7 Resources
A running MySQL server and physical storage.

5.1.1.8 References
No reference needed.

5.1.1.9 Processing
View business logic UML for a better understanding.

5.1.1.10 Data
INT UNSIGNED parenyCategoryId
INT UNSIGNED categoryId
VARCHAR(40) categoryName
INT UNSIGNED categoryColorCode

5.1.2 Thread

5.1.2.1 Type
Database table.

5.1.2.2 Purpose
Provide Thread data to the business logic layer.

5.1.2.3 Function
Provide Thread data to the business logic layer through standard MySQL queries.

5.1.2.4 Subordinates
None.

5.1.2.5 Dependencies
None.

5.1.2.6 Interfaces
The dataflow is controlled by MySQL.

5.1.2.7 Resources
A running MySQL server and physical storage.

5.1.2.8 References
No reference needed.

5.1.2.9 Processing
View business logic UML for a better understanding.

5.1.2.10 Data
INT UNSIGNED parentCategoryId
INT UNSIGNED userId
INT UNSIGNED threadId
VARCHAR(40) threadName
VARCHAR(1000) threadText

5.1.3 Replies

5.1.3.1 Type
Database table.

5.1.3.2 Purpose
Provide Replies data to the business logic layer.

5.1.3.3 Function
Provide Replies data to the business logic layer through standard MySQL queries.

5.1.3.4 Subordinates
None.

5.1.3.5 Dependencies
None.

5.1.3.6 Interfaces
The dataflow is controlled by MySQL.

5.1.3.7 Resources
A running MySQL server and physical storage.

5.1.3.8 References
No reference needed.

5.1.3.9 Processing
View business logic UML for a better understanding.

5.1.3.10 Data
INT UNSIGNED threadId
INT UNSIGNED replyId
INT UNSIGNED userId ´´
INT UNSIGNED time
INT UNSIGNED relevance
VARCHAR(1000) replyText

5.1.4 Users

5.1.4.1 Type
Database table.

5.1.4.2 Purpose
Provide Users data to the business logic layer.

5.1.4.3 Function
Provide Users data to the business logic layer through standard MySQL queries.

5.1.4.4 Subordinates
None.

5.1.4.5 Dependencies
None.

5.1.4.6 Interfaces
The dataflow is controlled by MySQL.

5.1.4.7 Resources
A running MySQL server and physical storage.

5.1.4.8 References
No reference needed.

5.1.4.9 Processing
View business logic UML for a better understanding.

5.1.4.10 Data
INT UNSIGNED userId
VARCHAR(200) profileDescription
DATE birthDate
INT limit
BOOL useDefaultColors

5.1.5 ThumbsUp

5.1.5.1 Type
Database table.

5.1.5.2 Purpose
Provide ThumbsUp data to the business logic layer.

5.1.5.3 Function
Provide ThumbsUp data to the business logic layer through standard MySQL queries.

5.1.5.4 Subordinates
None.

5.1.5.5 Dependencies
None.

5.1.5.6 Interfaces
The dataflow is controlled by MySQL.

5.1.5.7 Resources
A running MySQL server and physical storage.

5.1.5.8 References
No reference needed.

5.1.5.9 Processing
View business logic UML for a better understanding.

5.1.5.10 Data
INT UNSIGNED userId
INT UNSIGNED replyId

5.1.6 CustomCategoryColors

5.1.6.1 Type
Database table.

5.1.6.2 Purpose
Provide CustomCategoryColors data to the business logic layer.

5.1.6.3 Function
Provide CustomCategoryColors data to the business logic layer through standard MySQL queries.

5.1.6.4 Subordinates
None.

5.1.6.5 Dependencies
None.

5.1.6.6 Interfaces
The dataflow is controlled by MySQL.

5.1.6.7 Resources
A running MySQL server and physical storage.

5.1.6.8 References
No reference needed.

5.1.6.9 Processing
View business logic UML for a better understanding.

5.1.6.10 Data
INT UNSIGNED userId
INT UNSIGNED categoryId
INT UNSIGNED categoryColorCode

5.1.7 PrivateCategory

5.1.7.1 Type
Database table.

5.1.7.2 Purpose
Provide PrivateCategory data to the business logic layer.

5.1.7.3 Function
Provide PrivateCategory data to the business logic layer through standard MySQL queries.

5.1.7.4 Subordinates
None.

5.1.7.5 Dependencies
None.

5.1.7.6 Interfaces
The data flow is controlled by MySQL.

5.1.7.7 Resources
A running MySQL server and physical storage.

5.1.7.8 References
No reference needed.

5.1.7.9 Processing
View business logic UML for a better understanding.

5.1.7.10 Data
INT UNSIGNED userId
INT UNSIGNED privateCategoryParentId
INT UNSIGNED privateCategoryId
INT UNSIGNED categoryColorCode

5.1.8 LinkedThreads

5.1.8.1 Type
Database table.

5.1.8.2 Purpose
Provide LinkedThreads data to the business logic layer.

5.1.8.3 Function
Provide LinkedThreads data to the business logic layer through standard MySQL queries.

5.1.8.4 Subordinates
None.

5.1.8.5 Dependencies
None.

5.1.8.6 Interfaces
The dataflow is controlled by MySQL.

5.1.8.7 Resources
A running MySQL server and physical storage.

5.1.8.8 References
No reference needed.

5.1.8.9 Processing
View business logic UML for a better understanding.

5.1.8.10 Data
INT UNSIGNED userId
INT UNSGINED privateCategoryParentId
INT UNSIGNED linkedThreadId
VARCHAR(40) linkedThreadName

5.1.9 Notes

5.1.9.1 Type
Database table.

5.1.9.2 Purpose
Provide Notes data to the business logic layer.

5.1.9.3 Function
Provide Notes data to the business logic layer through standard MySQL queries.

5.1.9.4 Subordinates
None.

5.1.9.5 Dependencies
None.

5.1.9.6 Interfaces
The dataflow is controlled by MySQL.

5.1.9.7 Resources
A running MySQL server and physical storage.

5.1.9.8 References
No reference needed.

5.1.9.9 Processing
View business logic UML for a better understanding.

5.1.9.10 Data
INT UNSIGNED userId
INT UNSGINED privateCategoryParentId
VARCHAR(40) noteName
VARCHAR(1000) replyText

5.1.10 PrivateMessages

5.1.10.1 Type
Database table.

5.1.10.2 Purpose
Provide PrivateMessages data to the business logic layer.

5.1.10.3 Function
Provide PrivateMessages data to the business logic layer through standard MySQL queries.

5.1.10.4 Subordinates
None.

5.1.10.5 Dependencies
None.

5.1.10.6 Interfaces
The data flow is controlled by MySQL.

5.1.10.7 Resources
A running MySQL server and physical storage.

5.1.10.8 References
No reference needed.

5.1.10.9 Processing
View business logic UML for a better understanding.

5.1.10.10 Data
INT UNSIGNED senderUserId
INT UNSIGNED recieverUserId
VARCHAR(1000) messageText

5.2 Business Logic Components

5.2.1 Server

5.2.1.1 Type
Java class file.

5.2.1.2 Purpose
This class acts as a root class. It is the first class being initiated on client side and delivers all other objects that may be used.
Since all objects but the Server itself is requested through the Server, it is needed for any of the software requirements in the RURD to be functional.

5.2.1.3 Function
Holds references to the user profile as well as the private and public forums and PrivateMessaging.

5.2.1.4 Subordinates
Has 4 subordinates: Profile, PrivateMessageing, PrivateForum and PublicForum.

5.2.1.5 Dependencies
This class delivers Profile, PrivateMessage, PrivateForum and PublicForum.
It is not dependent on these classes in the sense that it will stop working if they do not work, however, if they work badly the Server will deliver bad objects.

5.2.1.6 Interfaces
When the application needs access to the user profile, public or private forum or the private messaging instance, a call to the server is made. The server is central to the application in this way.

5.2.1.7 Resources
No resources are needed, other than those necessary to initialize the components held by the server.

5.2.1.8 References
No reference needed.

5.2.1.9 Processing
View business logic UML for a better understanding.

5.2.1.10 Data
View business logic UML for a better understanding.

5.2.2 PrivateForum

5.2.2.1 Type
Java class file.

5.2.2.2 Purpose
Acts as a holder for the category tree that is the private forum.

5.2.2.3 Function
Holds references to the top categories in the private forum.

5.2.2.4 Subordinates
Has 1 subordinate: PrivateCategory.

5.2.2.5 Dependencies
Depends on the database to be able to access the categories in the private forum.

5.2.2.6 Interfaces
The class holds methods for accessing forum categories, both those in the root of the tree as well as all categories belonging to the private forum. It also offers a way to remove categories.

5.2.2.7 Resources
A database connection is needed for the class to fetch the wanted categories.

5.2.2.8 References
No reference needed.

5.2.2.9 Processing
View business logic UML for a better understanding.

5.2.2.10 Data
View business logic UML for a better understanding.

5.2.3 PublicForum

5.2.3.1 Type
Java class file.

5.2.3.2 Purpose
Acts as a holder for the category tree that is the public forum.

5.2.3.3 Function
Holds references to the top categories in the public forum.

5.2.3.4 Subordinates
Has 1 subordinate: Thread.

5.2.3.5 Dependencies
Depends on the database to be able to access the categories in the public forum.

5.2.3.6 Interfaces
The class holds methods for accessing forum categories, both those in the root of the tree as well as all categories belonging to the public forum. It also offers a way to remove categories.

5.2.3.7 Resources
A database connection is needed for the class to fetch the wanted categories.

5.2.3.8 References
No reference needed.

5.2.3.9 Processing
See business logic UML for a better understanding.

5.2.3.10 Data
View business logic UML for a better understanding.

5.2.4 PrivateCategory

5.2.4.1 Type
Java class file.

5.2.4.2 Purpose
Acts as a node in the private category tree.

5.2.4.3 Function
Holds the content of a category as well as references to its subcategories.

5.2.4.4 Subordinates
Has 2 subordinates: Note and LinkedThread.

5.2.4.5 Dependencies
Depends on the database to be able to access the subcategories and content of a category.

5.2.4.6 Interfaces
The class holds methods used to access content of a category in the private forum.

5.2.4.7 Resources
A database connection is needed for the class to fetch the wanted categories.

5.2.4.8 References
No reference needed.

5.2.4.9 Processing
View business logic UML for a better understanding.

5.2.4.10 Data
View business logic UML for a better understanding.

5.2.5 PublicCategory

5.2.5.1 Type
Java class file.

5.2.5.2 Purpose
Acts as a node in the public category tree.

5.2.5.3 Function
Holds the content of a category as well as references to its subcategories.

5.2.5.4 Subordinates
Has 2 subordinates: Note and LinkedThread.

5.2.5.5 Dependencies
Depends on the database to be able to access the subcategories and content of a category.

5.2.5.6 Interfaces
The class holds methods used to access content of a category in the public forum.

5.2.5.7 Resources
A database connection is needed for the class to fetch the wanted categories.

5.2.5.8 References
No reference needed.

5.2.5.9 Processing
View business logic UML for a better understanding.

5.2.5.10 Data
View business logic UML for a better understanding.

5.2.6 ForumItem

5.2.6.1 Type
Java class file.

5.2.6.2 Purpose
Acts as base class for Post and Category.

5.2.6.3 Function
Superclass.

5.2.6.4 Subordinates
Has 2 subordinates: Post and Category.

5.2.6.5 Dependencies
No dependencies.

5.2.6.6 Interfaces
Holds a method used to access the name of the forum item.

5.2.6.7 Resources
No resources are needed.

5.2.6.8 References
No reference needed.

5.2.6.9 Processing
View business logic UML for a better understanding.

5.2.6.10 Data
View business logic UML for a better understanding.

5.2.7 Post

5.2.7.1 Type
Java class.

5.2.7.2 Purpose
To contain a message in a public thread on forum.

5.2.7.3 Function
Post contains the message text, authors user ID, when it was posted and functions to manipulate them. A function to modify the message text is included as well.

5.2.7.4 Subordinates
None.

5.2.7.5 Dependencies
None.

5.2.7.6 Interfaces
A *Post* is a key part of *Notes*, *Thread* and *Replies* since it acts as container for the data and the other classes extends it to fulfill their respective purposes. In *Thread* it contains the initial message, in the reply it's the text itself and in *Note* it's the text as well.

5.2.7.7 Resources
None.

5.2.7.8 References
None.

5.2.7.9 Processing
A *Post* never does anything on its own, it's always under the control of another class such as *Reply* and *Thread*. Minimal functions to manipulate the data containing it contains.

5.2.8 Category

5.2.8.1 Type
Java class file.

5.2.8.2 Purpose
Acts as base class for PublicCategory and PrivateCategory.

5.2.8.3 Function
Superclass.

5.2.8.4 Subordinates
Has 2 subordinates: PublicCategory and PrivateCategory.

5.2.8.5 Dependencies
Depends on its superclass ForumItem.

5.2.8.6 Interfaces
Holds methods used to get and set color codes.

5.2.8.7 Resources
No resources are needed.

5.2.8.8 References
No reference needed.

5.2.8.9 Processing
View business logic UML for a better understanding.

5.2.8.10 Data
View business logic UML for a better understanding.

5.2.9 Profile

5.2.9.1. Type
Java class file.

5.2.9.2. Purpose
The profile contains personalized information about the user and is a container of personal info.

5.2.9.3 Function
Contains number of accesses per day restriction variable, whether the user wants to use custom colors or not, their birth date and a custom string that will be present on their profile page.

5.2.9.4 Subordinates
None

5.2.9.5 Dependencies
A *User* since it extends it in order to inherit the custom string, birth date and ID.

5.2.9.6 Interfaces
The Profile is used when the user customizes the color of the categories, when she updates her birth date and the custom string. By implication it is also used when category colors are drawn and whenever the forum profile is shown.

5.2.9.7 Resources
None.

5.2.9.8 References
None.

5.2.9.9 Processing
N/A.

5.2.9.10 Data
See Business Logic UML.

5.2.10 User

5.2.10.1 Type
Java class.

5.2.10.2 Purpose
Container for user personalization data.

5.2.10.3 Function
Acts as a container for a custom string, a birth date and a unique user ID.

5.2.10.4 Subordinates
Profile.

5.2.10.5 Dependencies
None.

5.2.10.6 Interfaces
Since *User* is a container class it is accessed every time the users customisable and personal information is needed.

5.2.10.7 Resources
None.

5.2.10.8 References
None.

5.2.10.9 Processing
N/A.

5.2.10.10 Data
See Business Logic UML.

5.2.11 Message

5.2.11.1 Type
Java class.

5.2.11.2 Purpose
Container class for private messages between users.

5.2.11.3 Function
Message is accessed when private message threads are created. It's an individual message in the thread.

5.2.11.4 Subordinates
None.

5.2.11.5 Dependencies
None.

5.2.11.6 Interfaces
Message functions as a container for messages and has the user ID of the sender as well as the recipient. It will not do any work on its own but will simply contain the necessary data for other classes.

5.2.11.7 Resources
None.

5.2.11.8 References
None.

5.2.11.9 Processing
None.

5.2.11.10 Data
See Business Logic UML.

5.2.12 PrivateMessaging

5.2.12.1 Type

Java class.

5.2.12.2 Purpose

A class that contains all functions needed to process messages between users.

5.2.12.3 Function

The relevant functions are to get messages, send messages, get a list of people with open conversations and to clean the message history.

5.2.12.4 Subordinates

Message.

5.2.12.5 Dependencies

None.

5.2.12.6 Interfaces

To get messages as well as send them means to extract the relevant parts of the message from the *Message* class and then process them. The user conversation list is a list of users that represents all open conversations.

5.2.12.7 Resources

None.

5.2.12.8 References.

None.

5.2.12.9 Processing

Conversations are lists of messages with messages with the alternating senders and recipients. The function will return a message, how many messages there are and where the first message is. All that is needed to send and receive a messages is either the current users ID or the recipients ID. Cleaning the conversation history is the same as deleting the messages that it contains.

5.2.12.10 Data

None. Everything is contained in the *Message* class.

5.2.13 Reply

5.2.13.1 Type
Java class.

5.2.13.2 Purpose
A class that contains all functions needed to post replies. Will itself act as a reply object when sent to the client.

5.2.13.3 Function
The relevant functionality is to have the variables and methods needed for a reply when it has been sent to the client to be presented.

5.2.13.4 Subordinates
Extends 5.2.7 Post.

5.2.13.5 Dependencies
In order for any object of type Reply to exist the component 5.2.24 Thread must exist.

5.2.13.6 Interfaces
A call to the thread class to create a reply is made and then that reply object gets stored in the database. Also a call to the thread could be made to retrieve reply objects to be sent and presented at the client.

5.2.13.7 Resources
None.

5.2.13.8 References.
None.

5.2.13.9 Processing
None.

5.2.13.10 Data
Consult the Business logic UML.

5.2.14 Thread

5.2.14.1 Type
Java class.

5.2.14.2 Purpose
A class that handles the content and possible actions on a forum thread.

5.2.14.3 Function
Contains methods for creating and deleting replies to a thread, as well as creating a link to the thread in the private forum.

5.2.14.4 Subordinates
Has 2 subordinates: Reply and LinkedThread.

5.2.14.5 Dependencies
Depends on Reply.

5.2.14.6 Interfaces
Used for creating replies to threads as well as modifying the thread.

5.2.14.7 Resources
Needs a database connection in order to create new replies and modify threads.

5.2.14.8 Refereces.
None.

5.2.14.9 Processing
See Business Logic UML.

5.2.14.10 Data
See Business Logic UML.

5.2.15 LinkedThread

5.2.15.1 Type
Java class.

5.2.15.2 Purpose
A class that contains extra features for the linked thread in the private forum.

5.2.15.3 Function
The relevant functionality is to have the variables and methods needed for a linked thread that the *5.2.24 Thread* class does not contain.

5.2.15.4 Subordinates
Extends 5.2.24 Thread.

5.2.15.5 Dependencies
In order for any object of type LinkedThread to exist the component *5.2.24 Thread* must exist.

5.2.15.6 Interfaces
A call to *5.2.4 PrivateCategory* class is made to create a Linked Thread and then that object gets stored in the database. Also a call to the PrivateCategory could be made to retrieve Linked Thread objects to be sent and presented at the client.

5.2.15.7 Resources
None.

5.2.15.8 References.
None.

5.2.15.9 Processing
None.

5.2.15.10 Data
View the Business logic UML.

5.1.16 Note

5.2.16.1 Type
Java class.

5.2.16.2 Purpose
Acts like a personal note in the private forum.

5.2.16.3 Function
Has the same functionality as the Post class.

5.2.16.4 Subordinates
No subordinates.

5.2.16.5 Dependencies
Depends on Post.

5.2.16.6 Interfaces
See Post.

5.2.16.7 Resources
See Post.

5.2.16.8 Refereces.
None.

5.2.16.9 Processing
See Business Logic UML.

5.2.16.10 Data
See Business Logic UML.

5.3 Client Structure Components

5.3.1 Engine

5.3.1.1 Type
Java class file.

5.3.1.2 Purpose
This class represents the core of the client and is needed for any of the software requirements in the RURD to be functional.

5.3.1.3 Function
Extends the class android.app.Application so it will be accessible from anywhere in the client. This means that this component is also responsible for the initiation of the client aswell as handling different standard Android functions such as memory clearing. The component will hold objects of such kind that they are needed from anywhere within the client.

5.3.1.4 Subordinates
Has no direct subordinate. You can say that all other client components are subordinates.

5.3.1.5 Dependencies
Since this is the core of the client, nothing in the client is really needed for just this class to work. However for the component to achieve its goal and purpose (which is all the software requirements in the RURD) all other classes in the client must be functional.

5.3.1.6 Interfaces
Initialization of the application will begin with the *Engine* class. Whenever another class in the client needs something from our singleton objects, a call to the application (with a cast to *Engine*) is made and the needed methods are called or data retrieved. This means that the *Engine* class basically acts as a bridge between variable holding singleton objects and the active activity/fragment classes during runtime.

5.3.1.7 Resources
To fully work the server containing the database must be running. Also the business logic must be implemented and functional. This is basic demand for all the classes in the client and will not be repeated for the remaining components.

5.3.1.8 References
No reference needed.

5.3.1.9 Processing
View client UML for a better understanding.

5.3.1.10 Data
View client UML for a better understanding.

5.3.2 PublicForum

5.3.2.1 Type
Java class file.

5.3.2.2 Purpose
Responsible of handling methods and lasting variables that only concerns the public forum and are accessible from anywhere within the application. Strongly connected to the requirement *3.1.1.2 Public Forum* in the RURD.

5.3.2.3 Function
The component is a singleton class and functions as a globally accessible variable and method holder regarding the public forum. Its object will be contained in the core component *5.3.1 Engine.*

5.3.2.4 Subordinates
Extends component: *5.3.4 Forum.*

5.3.2.5 Dependencies
Component: *5.3.4 Forum.*

5.3.2.6 Interfaces
Through the *5.3.1 Engine* class (that holds the object of this component) calls will come from basically anywhere within the application when variables or methods regarding the public forum are needed.

5.3.2.7 Resources
Same as *5.3.1 Engine.*

5.3.2.8 References
No reference needed.

5.3.2.9 Processing
View client UML for a better understanding.

5.3.2.10 Data
View client UML for a better understanding.

5.3.3 PrivateForum

5.3.3.1 Type
Java class file.

5.3.3.2 Purpose
Responsible of handling methods and lasting variables that only concerns the private forum and are accessible from anywhere within the application. Strongly connected to the requirement *3.1.1.3 Private Forum* in the RURD.

5.3.3.3 Function
The component is a singleton class and function as a global accessible variable and method holder regarding the private forum. Its object will be contained in the core component *5.3.1 Engine.*

5.3.3.4 Subordinates
Extends component: *5.3.4 Forum.*

5.3.3.5 Dependencies
Component: *5.3.4 Forum.*

5.3.3.6 Interfaces
Through the *5.3.1 Engine* class (that holds the object of this component) calls will come from basically anywhere within the application when variables or methods regarding the private forum are needed.

5.3.3.7 Resources
Same as *5.3.1 Engine.*

5.3.3.8 References
No reference needed.

5.3.3.9 Processing
View client UML for a better understanding.

5.3.3.10 Data
Check client UML for a better understanding.

5.3.4 Forum

5.3.4.1 Type
Java class file.

5.3.4.2 Purpose
Responsible of handling methods and lasting variables that are accessible from anywhere within the application concerning both the private forum and public forum . Strongly connected to the requirement *3.1.1.3 Private Forum* in the RURD aswell as *3.1.1.2 Public Forum.*

5.3.4.3 Function
The component is a extended class that functions as a globally accessible variable and method holder for both the forums. Its extended objects will be contained in the core component *5.3.1 Engine.*

5.3.4.4 Subordinates
No subordinates.

5.3.4.5 Dependencies
Component: *5.3.4 Forum.*

5.3.4.6 Interfaces
Through the *5.3.1 Engine* class (that holds the object of this component) calls will come from basically anywhere within the application when variables or methods regarding the private forum are needed.

5.3.4.7 Resources
Same as *5.3.1 Engine.*

5.3.4.8 References
No reference needed.

5.3.4.9 Processing
View client UML for a better understanding.

5.3.4.10 Data
Check client UML for a better understanding.

5.3.5 PersonalMessages

5.3.5.1 Type
Java class file.

5.3.5.2 Purpose
Somewhere the users message log must be contained and that is what this singleton class is for. This component is important for the requirement *3.1.4.4 Chat Function* in the RURD.

5.3.5.3 Function
The component is a singleton class and function as a global accessible variable and method holder regarding a user´s private messages. Its object will be contained in the core component *5.3.1 Engine.*

5.3.5.4 Subordinates
No subordinates.

5.3.5.5 Dependencies
No dependencies.

5.3.5.6 Interfaces
Through the *5.3.1 Engine* class (that holds the object of this component) calls will come from basically anywhere within the application when variables or methods regarding the messages are needed.

5.3.5.7 Resources
Same as *5.3.1 Engine*

5.3.5.8 References
No reference needed.

5.3.5.9 Processing
View client UML for a better understanding.

5.3.5.10 Data
View client UML for a better understanding.

5.3.6 Account

5.3.6.1 Type
Java class file.

5.3.6.2 Purpose
When the user authenticates themselves the gathered user information must be kept somewhere during application runtime. This component handles just that. This class is to some extent needed for the requirement *3.1.2.3 Login functionality* in the RURD to achieve its purpose.

5.3.6.3 Function
The component is a singleton class and function as a global accessible variable and method holder regarding a user's account information. Its object will be contained in the core component *5.3.1 Engine.*

5.3.6.4 Subordinates
No subordinates.

5.3.6.5 Dependencies
No dependencies.

5.3.6.6 Interfaces
Through the *5.3.1 Engine* class (that holds the object of this component) calls will come from basically anywhere within the application when variables or methods regarding the user´s account are needed.

5.3.6.7 Resources
Same as *5.3.1 Engine*

5.3.6.8 References
No reference needed.

5.3.6.9 Processing
View client UML for a better understanding.

5.3.6.10 Data
View client UML for a better understanding.

5.3.7 LoginActivity

5.3.7.1 Type
Java class file.

5.3.7.2 Purpose
This class is responsible for handling the authentication regarding the user. The component will solve software requirement *3.1.2.3 Login functionality in the RURD.*

5.3.7.3 Function
The component is an activity class and will be the first view presented to the user (unless some sort of automated authentication speeds through this activity). The activity handles both the logic regarding authentication and the graphics going with it.

5.3.7.4 Subordinates
No subordinates.

5.3.7.5 Dependencies
The user needs a Facebook Account.
Since we need to store the users information somewhere the *5.3.6 Account* object must exist.

5.3.7.6 Interfaces
The *5.3.1 Engine* class will activate this activity at startup and ask the user to authenticate. When the user is approved his or hers information will be stored in the *5.3.6 Account* object and *5.3.1 Engine* will load another activity and this component will not be reached again unless the user terminates and restarts the application.

5.3.7.7 Resources
Same as *5.3.1 Engine*

5.3.7.8 References
No reference needed.

5.3.7.9 Processing
View client UML for a better understanding.

5.3.7.10 Data
View client UML for a better understanding.

5.3.8 HomeActivity

5.3.8.1 Type
Java class file.

5.3.8.2 Purpose
This class is supposed to act as the main menu of the application. The purpose of this component is to give the user a good overview and starting point of the application. The component´s purpose is strongly connected to the requirement *3.1.3.1 Easy to Navigate* in the RURD.

5.3.8.3 Function
The component is an activity class and will be the main view presented to the user after logging in and pressing the “home” button. The activity acts as a navigation panel for the application and presents them in an appealing fashion.

5.3.8.4 Subordinates
No subordinates.

5.3.8.5 Dependencies
Everything that this activity navigates to must be implemented of course.

5.3.8.6 Interfaces
When the user clicks the “back” button he/she will eventually end up at this activity. Also if the “home” button is pressed the user will end up at this activity. The component has no variables or objects in it, it is just the applications main menu.

5.3.8.7 Resources
Same as *5.3.1 Engine*

5.3.8.8 References
No reference needed.

5.3.8.9 Processing
View client UML for a better understanding.

5.3.8.10 Data
View client UML for a better understanding.

5.3.9 PublicForumActivity

5.3.9.1 Type
Java class file.

5.3.9.2 Purpose
This is the activity handling the fragments regarding the public forum. It´s purpose is to display the different fragment elements in a logical and visual appealing way. This component is connected to the software requirements *3.1.3.2 Clutter-free design* and *3.1.1.2 Public Forum.*

5.3.9.3 Function
This is an activity class that will not itself present much for the user but instead is responsible of showing fragments that acts as smaller activities, regarding the public forum.

5.3.9.4 Subordinates
Expands 5.3.13 PublicForumFragment.
Extends 5.3.12 ForumActivity.

5.3.9.5 Dependencies
No dependencies.

5.3.9.6 Interfaces
Like stated in *Function* the component itself acts as a closer/opener for the direct and indirect connected fragments. It itself has no data flowing through it.

5.3.9.7 Resources
Same as *5.3.1 Engine*

5.3.9.8 References
No reference needed.

5.3.9.9 Processing
View client UML for a better understanding.

5.3.9.10 Data
View client UML for a better understanding.

5.3.11 PrivateForumActivity

5.3.11.1 Type
Java class file.

5.3.11.2 Purpose
This is the activity handling the fragments regarding the private forum. It´s purpose is to display the different fragment elements in a logical and visual appealing way. This component is connected to the software requirements *3.1.3.2 Clutter-free design* and *3.1.1.3 Private Forum.*

5.3.11.3 Function
This is an activity class that will not itself present much for the user but instead is responsible of showing fragments that acts as smaller activities, regarding the private forum.

5.3.11.4 Subordinates
Expands 5.3.14 PrivateForumFragment.
Extends 5.3.12 ForumActivity.

5.3.11.5 Dependencies
No dependencies.

5.3.11.6 Interfaces
Like stated in *Function* the component itself acts as a closer/opener for the direct and indirect connected fragments. It itself has no data flowing through it.

5.3.11.7 Resources
Same as *5.3.1 Engine*

5.3.11.8 References
No reference needed.

5.3.11.9 Processing
View client UML for a better understanding.

5.3.11.10 Data
View client UML for a better understanding.

5.3.12 ForumActivity

5.3.12.1 Type
Java class file.

5.3.12.2 Purpose
This is the activity handling the fragments regarding both the private and public forum. It´s purpose is to display the different fragment elements in a logical and visual appealing way. This component is mainly connected to the software requirements *3.1.3.2 Clutter-free design.*

5.3.12.3 Function
This is an activity class that contains local methods and variables that should be accessible for both the *5.3.9 PublicForumActivity* and *5.3.11 PrivateForumActivity.*

5.3.12.4 Subordinates
No subordinates.

5.3.12.5 Dependencies
No dependencies.

5.3.12.6 Interfaces
Check *5.3.9 PublicForumActivity* and *5.3.11 PrivateForumActivity.*

5.3.12.7 Resources
Same as *5.3.1 Engine*

5.3.12.8 References
No reference needed.

5.3.12.9 Processing
View client UML for a better understanding.

5.3.12.10 Data
View client UML for a better understanding.

5.3.13 PublicForumFragment

5.3.13.1 Type
Java class file.

5.3.13.2 Purpose
The fragment responsible for showing items and navigation within the public forum. Important class for the software requirement *3.1.1.2 Public Forum* in the RURD.

5.3.13.3 Function
This class functions as a fragment connected to an activity and displays items such as categories and threads. Its main function is to show all items connected to the public forum in a sensible and visually appealing way.

5.3.13.4 Subordinates
Extends *5.3.15 ForumFragment.*
Expands *5.3.16 WriteThreadFragment.*

5.3.13.5 Dependencies
The activity *5.3.9 PublicActivity* must exist since this is the base on which the fragment will lay upon.

5.3.13.6 Interfaces
The data flowing through this class are categories and threads gathered from the database. Whenever an item is opened, new items are loaded from the database and are then presented in the fragment.

5.3.13.7 Resources
Same as *5.3.1 Engine*

5.3.13.8 References
No reference needed.

5.3.13.9 Processing
View client UML for a better understanding.

5.3.13.10 Data
View client UML for a better understanding.

5.3.14 PrivateForumFragment

5.3.14.1 Type
Java class file.

5.3.14.2 Purpose
The fragment responsible for showing items and navigation within the private forum. Important class for the software requirement *3.1.1.3 Private Forum* in the RURD.

5.3.14.3 Function
This class functions as a fragment connected to an activity and displays items such as categories and threads. Its main function is to show all items connected to the public forum in a sensible and visually appealing way.

5.3.14.4 Subordinates
Extends *5.3.15 ForumFragment.*
Expands *5.3.17 WritePostFragment.*

5.3.14.5 Dependencies
The activity *5.3.11 PrivateForumActivity* must exist since this is the base on which the fragment will lay upon.

5.3.14.6 Interfaces
The data flowing through this class are categories, notes and linked threads gathered from the database. Whenever an item is opened, new items are loaded from the database and are then presented in the fragment.

5.3.14.7 Resources
Same as *5.3.1 Engine*

5.3.14.8 References
No reference needed.

5.3.14.9 Processing
View client UML for a better understanding.

5.3.14.10 Data
View client UML for a better understanding.

5.3.15 ForumFragment

5.3.15.1 Type
Java class file.

5.3.15.2 Purpose
The fragment responsible for showing items and navigation within the private and public forum. Important class for the software requirement *3.1.1.3 Private Forum* and *3.1.1.2 Public Forum* in the RURD.

5.3.15.3 Function
This class functions as a common fragment for the *5.3.13 PublicForumFragment* and *5.3.14 PrivateForumFragment*. Like the named components, the function is to show all items in a sensible and visually appealing way.

5.3.15.4 Subordinates
Expands *5.3.18 ThreadFragment.*
Expands *5.3.19 ProfileFragment.*

5.3.15.5 Dependencies
The fragments *5.3.13 PublicForumFragment* and *5.3.14 PrivateForumFragment* must of course be functional for this component to fully function.

5.3.15.6 Interfaces
Check *5.3.13 PublicForumFragment* and *5.3.14 PrivateForumFragment*.

5.3.15.7 Resources
Same as *5.3.1 Engine*

5.3.15.8 References
No reference needed.

5.3.15.9 Processing
View client UML for a better understanding.

5.3.15.10 Data
View client UML for a better understanding.

5.3.16 WriteThreadFragment

5.3.16.1 Type
Java class file.

5.3.16.2 Purpose
The purpose of this class is to handle the writing of new threads within the public forum. Makes sure that the software requirement *3.1.2.1 Thread creation functionality* in the RURD is met.

5.3.16.3 Function
This component function as a fragment that displays the area on which the user can create a new thread for the public forum.

5.3.16.4 Subordinates
Extends *5.3.17 WritePostFragment.*

5.3.16.5 Dependencies
Needs *5.3.13 PublicForumFragment* to work since the call to create a thread will be made from that fragment.

5.3.16.6 Interfaces
When opened, the user gives input such as text and a headline. When posted a call to the database is made to store the newly created thread.

5.3.16.7 Resources
Same as *5.3.1 Engine*

5.3.16.8 References
No reference needed.

5.3.16.9 Processing
View client UML for a better understanding.

5.3.16.10 Data
View client UML for a better understanding.

5.3.17 WritePostFragment

5.3.17.1 Type
Java class file.

5.3.17.2 Purpose
The purpose of this class is to handle the writing of new replies and notes within the public and private forum. Mainly makes sure that the software requirement *3.1.2.2 Commenting functionality* in the RURD is met.

5.3.17.3 Function
This component function as a fragment that displays the area on which the user can create a new reply or note for the public respektive private forum.

5.3.17.4 Subordinates
No subordinates.

5.3.17.5 Dependencies
Needs at least *5.3.14 PrivateForumFragment* and/or *5.3.18 ThreadFragment* to work since the call to create a reply/note will be made from those fragments.

5.3.17.6 Interfaces
When opened, the user gives input such as text. When posted a call to the database is made to store the newly created post, either as a note or as a reply depending on if the user was in the private forum or in the public forum when the post was made.

5.3.17.7 Resources
Same as *5.3.1 Engine*

5.3.17.8 References
No reference needed.

5.3.17.9 Processing
View client UML for a better understanding.

5.3.17.10 Data
View client UML for a better understanding.

5.3.18 ThreadFragment

5.3.18.1 Type
Java class file.

5.3.18.2 Purpose
Responsible for showing a thread with its replies. Important for the software requirement *3.1.4.3 Visually Engaging* aswell as *3.1.4.1 Structured Forum* in the RURD.

5.3.18.3 Function
Displays items in a list form. The thread post will be on top followed by replies to that post. Sorting will be possible to get the replies in different orders.

5.3.18.4 Subordinates
Expands *5.3.17 WritePostFragment.*
Expands *5.3.19 ProfileFragment.*

5.3.18.5 Dependencies
The component *5.3.15 ForumFragment* is needed since all calls to show a thread will come from that fragment.

5.3.18.6 Interfaces
When the user clicks on a thread, this fragment will expand. When the user makes a reply the database will be called and the reply will then be stored. Other database calls will also be possible.

5.3.18.7 Resources
Same as *5.3.1 Engine*

5.3.18.8 References
No reference needed.

5.3.18.9 Processing
View client UML for a better understanding.

5.3.18.10 Data
View client UML for a better understanding.

5.3.19 ProfileFragment

5.3.19.1 Type
Java class file.

5.3.19.2 Purpose
The purpose of this component is to show some user´s profile in a visually appealing manner. The component is not strongly connected to any software requirement but since it is important to know who your friends are it is in some way connected to *3.1.1.1 Connecting People with ADHD* in the RURD.

5.3.19.3 Function
This class functions as a profile viewing fragment and is a stepping block to sending a message to someone. If the user is checking its own profile then he/she can change some of the information.

5.3.19.4 Subordinates
Expands *PersonalMessageFragment.*

5.3.19.5 Dependencies
Needs at least *5.3.15 Forumfragment* or *5.3.18 ThreadFragment* or *5.3.20 Settingsfragment* to be functional in some way since calls to this fragment are made from there.

5.3.19.6 Interfaces

5.3.19.7 Resources
Same as *5.3.1 Engine*

5.3.19.8 References
No reference needed.

5.3.19.9 Processing
View client UML for a better understanding.

5.3.19.10 Data
View client UML for a better understanding.

5.3.20 SettingsFragment

5.3.20.1 Type
Java class file.

5.3.20.2 Purpose
This component is needed since the user must be able to change different settings to the application. This component is connected to the software requirement *3.1.4.1 Structured Forum.*

5.3.20.3 Function
The functionality is basic changing of different settings that we allow on the application.

5.3.20.4 Subordinates
Expands *5.3.19 ProfileFragment.*

5.3.20.5 Dependencies
To be able to access this fragment the *5.3.22 BaseActivity* must be functional since the calls are made from this activity.

5.3.20.6 Interfaces
This fragment is the application's main option screen and will be accessible from any activity except the *5.3.7 LoginActivity.* Changing values here will make calls to the database that the user have new settings.

5.3.20.7 Resources
Same as *5.3.1 Engine*

5.3.20.8 References
No reference needed.

5.3.20.9 Processing
View client UML for a better understanding.

5.3.20.10 Data
View client UML for a better understanding.

5.3.21 PersonalMessageFragment

5.3.21.1 Type
Java class file.

5.3.21.2 Purpose
This component exists to give the user a nice view of the send and recieved messages. It also makes it possible for the user to write messages on its own. This class is strongly connected to the requirement *3.1.4.4 Chat Function* in the RURD.

5.3.21.3 Function
The fragment function like an email or chat service to the application between different users.

5.3.21.4 Subordinates
No subordinates.

5.3.21.5 Dependencies
Needs either *5.3.19 ProfileFragment* or *5.3.22 BaseActivity* since calls are made from these components to access this fragment.

5.3.21.6 Interfaces
The data flow of this component will be take form as database calls to and from the server, sending and receiving personal messages.

5.3.21.7 Resources
Same as *5.3.1 Engine*

5.3.21.8 References
No reference needed.

5.3.21.9 Processing
View client UML for a better understanding.

5.3.21.10 Data
View client UML for a better understanding.

5.3.22 BaseActivity

5.3.22.1 Type
Java class file.

5.3.22.2 Purpose
The purpose of this activity class is to have an activity extended by all other activities so that *5.3.20 SettingsFragment* can be accessible from anywhere. This class is strongly connected to the software requirements under *3.1.3 Graphic Requirements* in the RURD.

5.3.22.3 Function
Like stated in purpose the class is extended by all other activities and contain any common method and/or variable between the activities. The most important function is to make the *5.3.20 SettingsFragment* accessible by clicking on "Settings".

5.3.22.4 Subordinates
Expands *5.3.20 SettingsFragment.*
Expands *5.3.21 PersonalMessageFragment.*

5.3.22.5 Dependencies
In order to function, at least one of the other activities that extends this activity must be functional.

5.3.22.6 Interfaces
From a menu on the top the user can access both the settings and personal message no matter where he is situated in the application after he has been authenticated. Other then offering navigation possibilities the component has to data flow by itself.

5.3.22.7 Resources
Same as *5.3.1 Engine*

5.3.22.8 References
No reference needed.

5.3.22.9 Processing
View client UML for a better understanding.

5.3.22.10 Data
View client UML for a better understanding.

6. Feasibility and Resource Estimates

As we have multiple programmers we are able to divide the necessary components into three types, following the Three-Tiered model. As such, we declare three separate orderings, as each tier is necessary for the completion of the whole project.

Priority specifies how early we should start on a component. Need specifies how important it is. When two components priority and need are identical, we've specified the order after careful consideration.

Database

Component	**Priority**	**Need**	**Order (descending)**
PublicCategory	High	Budget	1
Thread	High	Budget	2
Replies	High	Budget	3
Users	High	Budget	4
PrivateCategory	High	Budget	5
PrivateMessages	Medium	Budget	6
LinkedThreads	Medium	Standard	7
ThumbsUp	Low	Standard	8
Notes	Low	Standard	9
CustomCategoryColors	Low	Deluxe	10

Business Logic

Component	Priority	Need	Order (descending)
ForumItem	High	Budget	1
Category	High	Budget	2
Post	High	Budget	3
Reply	High	Budget	4
Thread	High	Budget	5
PublicCategory	High	Budget	6
PublicForum	High	Budget	7
Server	High	Budget	8
User	High	Budget	9
Profile	High	Budget	10
PrivateCategory	High	Budget	11
PrivateForum	High	Budget	12
Message	High	Budget	13
PrivateMessaging	High	Budget	14
LinkedThread	Medium	Standard	15
Note	Low	Deluxe	16

Client side

Component	Priority	Need	Order (descending)
Engine	High	Budget	1
PublicForum	High	Budget	2
Forum	High	Budget	3
PublicForumActivity	High	Budget	4
ForumActivity	High	Budget	5
PublicForumFragment	High	Budget	6
ForumFragment	High	Budget	7
BaseActivity	High	Budget	8
ThreadFragment	High	Budget	9
WriteThreadFragment	High	Budget	10
WritePostFragment	High	Budget	11
Account	High	Budget	12
ProfileFragment	High	Budget	13
PrivateForum	High	Budget	14
LoginActivity	High	Budget	15
HomeActivity	High	Budget	16
PrivateForumActivity	High	Budget	17
PrivateForumFragment	High	Budget	18
SettingsFragment	Medium	Budget	19
PersonalMessageFragment	Low	Deluxe	20
PersonalMessages	Low	Deluxe	21

Potential risks

Veryday not giving us a (proper) server:
Impact: If we are unable to receive a proper server from our business client, we will simply have to proceed with our own, temporary alternative. Naturally, without a server the application won't go viral. Indeed, so far Veryday has been unable to provide us with a server, and as such, we are using our own server instead.
UI does not match target user group
Impact: Our application and forum is created for a special target group and if the UI and GUI are not suited for them - they won't use it. If we create a bad UI we have to get expertise help from our client, although this risk is very hard to measure.
Programmers aren't able to implement all functionalities
Impact: Depending on the functionality this could be severe.
Response: If this occurs it is important that we have followed the priority schedule so that our least important functionalities are not implemented.

Work to be done:
Database
- Create tables
- Fill tables with dummy data
- Create all relevant queries for the application logic layer

Inter-Process Communication Layer
- Get RPC running
- Get to run over SSL
- Get Facebook Authentication to act as the authentication mechanism

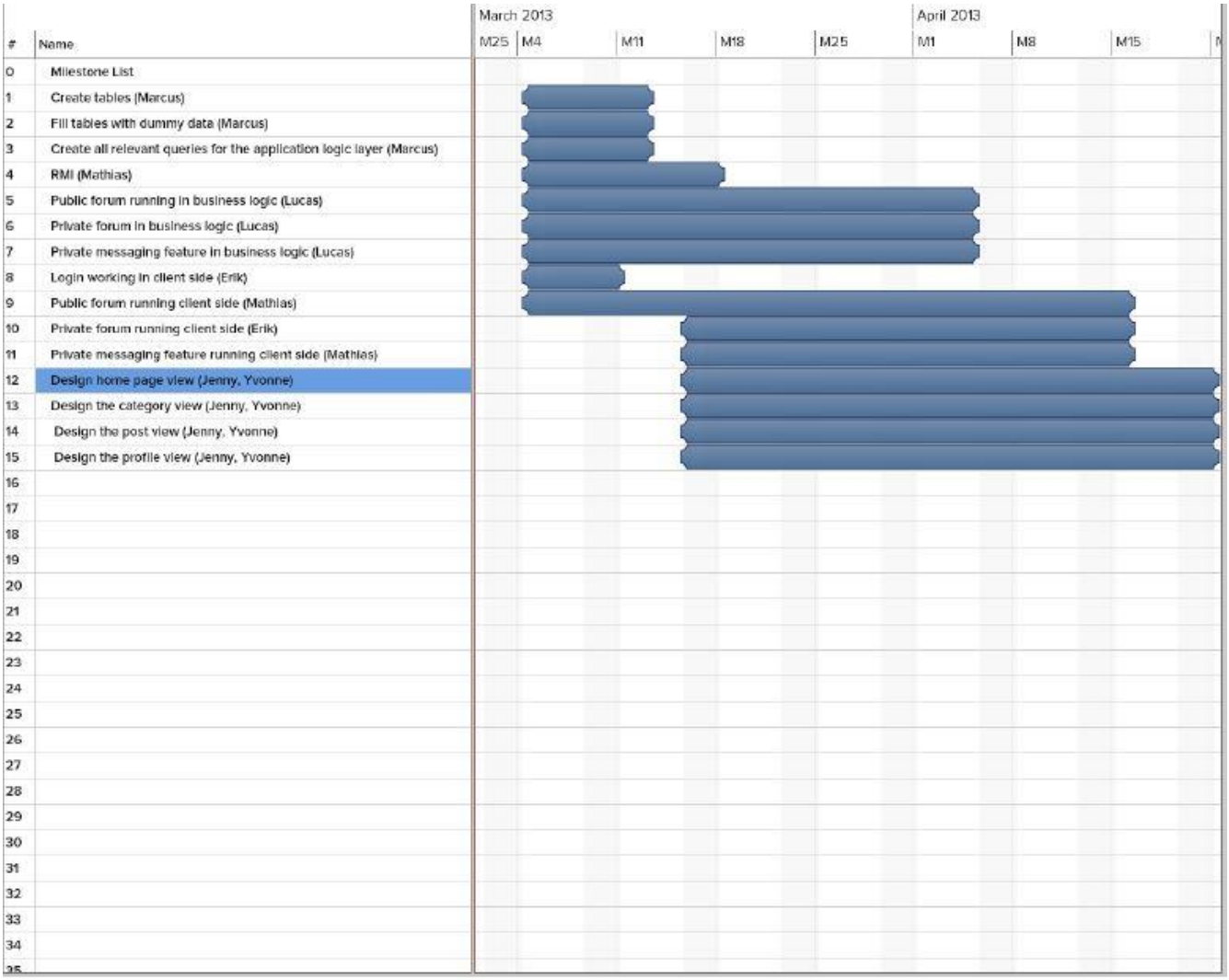
Implement business logic
- Get the public forum running
- Get the private forum running
- Get the private messaging feature running

Implement non-design code on client side
- Get login working
- Get the public forum running
- Get the private forum running
- Get the private messaging feature running

Implement design
- Design the home page view
- Design the category view
- Design the post view
- Design the profile view

Make Veryday set up a server with Java runtime environment and MySQL database.
Test the server, prep the database and then start to implement business layers in order of priority.

Gantt Flow Chart of Future Project Tasks:



7. User Requirements vs Components Traceability Matrix

1. *3.1.1.1 Connecting People with ADHD*
2. *3.1.2.1 Thread creation functionality*
3. *3.1.1.2 Public Forum*
4. *3.1.1.3 Private Forum*
5. *3.1.2.2 Commenting functionality*
6. *3.1.2.3 Login functionality*
7. *3.1.2.4 Admin functionality*
8. *3.1.2.5 Search functionality*
9. *3.1.2.6 Storage Requirements*
10. *3.1.4.1 Structured Forum*
11. *3.1.4.2 Interaction between Public and Private Forums*
12. *3.1.4.4 Chat Function*

	1	2	3	4	5	6	7	8	9	10	11	12
Database												
PublicCategories	X		X					X	X			
Thread	X	X	X		X			X	X		X	
Replies	X	X	X		X			X	X		X	
Users	X					X	X		X			
ThumbsUp			X						X			
CustomCategoryColors			X	X					X			
PrivateCategories				X				X	X			
LinkedThreads		X		X				X	X		X	
Notes				X	X			X	X			
PrivateMessages	X				X				X			
Business Logic												
Server	X		X	X	X					X	X	
PrivateForum				X	X							
PublicForum	X		X		X			X				
PrivateCategory		X		X					X			
PublicCategory		X	X					X	X			
ForumItem			X	X	X				X	X		

Post	X		X	X	X				X	X		
Category			X	X				X	X			
Profile	X					X			X			
User	X					X	X		X		X	
Message	X								X			X
PrivateMessageing	X											
Reply	X		X		X			X	X		X	
Thread	X	X	X		X		X	X	X		X	
LinkedThread		X		X					X		X	
Note				X	X				X			
Client structure components												
Engine			X	X	X			X	X		X	
PublicForum		X	X		X			X	X			
PrivateForum				X	X				X			
Forum		X	X	X	X			X	X		X	
PersonalMessages		X							X			
Account		X				X			X			X
LoginActivity						X						
HomeActivity								X		X	X	
PublicForumActivity	X	X	X		X			X				
PrivateForumActivity	X			X	X							
ForumActivity	X	X	X	X	X			X			X	
PublicForumFragment	X	X	X		X			X				
PrivateForumFragment	X			X	X							
ForumFragment	X	X	X	X	X			X			X	
WriteThreadFragment	X	X	X		X							
WritePostFragment		X	X	X	X							

ThreadFragment	X	X			X			X			X	
ProfileFragment		X										
SettingsFragment						X					X	
PersonalMessageFragment		X										
BaseActivity	X	X	X	X	X	X	X	X	X	X	X	X

8. Appendices

8.1 Meeting 2013-01-28, 1715-1800

Present:
Kim Malmros,
Jesper Norberg,
Yvonne Le,
Martin Pettersson,
Marcus Nordström,
Lucas Wiener

Moderator: Jesper
Secretary: Kim

1. Programmers
The programmers will develop the app and the rest of us will test it and help out with programming tasks in case of special circumstances or if someone feels like it. All Git tasks connected with the budget version should be put up by the programming group. As a result it follows naturally that the programming group sets up the tasks flowchart at least a significant portion of it. The rest of the group gives their input as the work progresses.
A facebook post will be made to get people started with Git and point out guides for the same.

2. Task
All members should get acquainted with Github - the website, not the program - and at least toy a bit with Git itself until Monday. Everyone should download and run the available code. Jennie contacts Veryday about the server they are to provide since it is an essential tool in the development process.

3. RURD
The final touches on the revised URD will be made by Kim and Jesper today.

4. Misc
It is important that people attend the lecture on the 8th since Jesper is unable to attend.

8.2 Notes from meeting, 2013-02-05, 1715-17

Present:
Yvonne Le,
Kim Malmros,
Jesper Norberg,
Marcus Nordström,
Jennie Olsson,
Martin Pettersson,
Lucas Wiener,
Leo Yu
Erik Odenman,
Mathias Lindblom

Moderator: Jesper Norberg
Secretary: Kim Malmros

1. Feedback from presentation or rURD
Veryday should be more active in the decision making and coding of the project. They have 20 hours per month to devote to the project so those should be utilized. Issues on Github for example. They had no comments on the URD other than "pretty much OK", we should have pressed for more detailed replies with leading questions on specific parts.

2. Programmers
All broad issues are up on Github. The UML should be looked over and then used as a plan to the program structure. The programmers should fill in the ADD where relevant with the things that they are responsible for. It doesn't need to be pretty, just functional as the report writer will polish it later on.

3. Veryday communication
Ask about the server. Are we supposed to be able to support only the app or are other clients to be expected?

4. Misc
The programmers and design folks should meet up and talk shop ASAP to get the ball rolling.. Berghs should be told that the screen dimension will be flexible but if they design for the iPhone format that is OK. APPDHD is the most hilarious app name proposed yet.

8. 3 Notes from meeting nr 3, 2013-02-18, 15:00-16.00

Present:
Yvonne Le,
Jesper Norberg,
Marcus Nordström,
Martin Pettersson,
Lucas Wiener,
Erik Odenman

Absent:
Kim Malmros,
Jennie Olsson

Moderator: Marcus Nordström
Secretary: Marcus Nordström

1. Set up server tomorrow at Martins or Mathias place.

2. Server from Veryday, very important.

3. Discussion on and standards to use in client/server communication, probably google or facebook authentication and JSON-RPC.

4. Get a design from the design group in collaboration with Bergs's.

5. Possibly assign one person as responsible for GIThub.

6. Meeting minutes from Kim, Jesper will help motivate him. Kim can test issues, etc.

7. Tabels for the ADD should be completed for tomorrow.

8. Martin needs help for the system design. Marcus has helped him so far.

9. Martin fixes introduction and abstract when the other parts of the essay are completed.

10. Jesper will proof read the essay and help Martin with the construction of it, and make sure progress is made.