# System Design Document - Yellow Project

William Diedrichsen Marstrand, Dennis Thinh Tan Nguyen,
Jacob Mullit Miniche, Thor Valentin Olesen

October 22, 2015

## 1 Introduction

## 1.1 Purpose of the System

In **AUTOSYS**, the main client is the **SystemReviewClient** who needs to set up a **STUDY** consisting of one or several **PHASES**. A **PHASE** will consist of one or more **TASKS** which need to be completed before moving on to the next **PHASE**. The **StudyParticipants** can work in different **ROLES** which we anticipate will be either as **REVIEWER** or **VALIDATOR**. Also the **StudyParticipants** will be working together on a **STUDY** in **TEAMS**, but multiple teams will not be working on the same **STUDY**. Furthermore **TEAMS** working on the same **STUDY** will be considered as one big **TEAM**.

## 1.2 Design Goals

- **Ease of use**:
  Goal: It should be relatively easy to setup a study configuration because it makes up the foundation that all study work processes rely on.

  The end user may have a low level of computer expertise potentially resulting in the wrong setup of a study. This can happen because the user cannot find or access the ressources required for setting up a study or they become frustrated if they have to go through many windows. However, these usability traits should not compromise with the system functionalities. The system must still be sufficiently complex in order to provide a variety of ways to setup the configuration. A primary focus on usability traits has been chosen due to the scope and time span of the project. This design goal is a refinement of the non-functional requirement "usability" in Requirement Analysis Document (section 3.3.1).

- **High Reliability**:
  Goal: The server and study configuration ui must be reliable, handle system failures and wrong user input.

  The server should handle system failures (e.g. exceptions or network failures). Due to the client-server architecture, it is important that the server can automatically reboot upon system failures. As a result, other clients can continuously access the server. Also, the server should ensure that incomplete data transfers are resumed in case of network failures. Further, the Study Configuration UI should handle invalid user input and ensure the input is only sent to the server when correct. Finally, the server should have a backup of the database in case of corruption. These decisions ensure that the work of the end user is handled properly and the server is available when needed. This design goal is a refinement of the non-functional requirement "high reliability" in Requirement Analysis Document (section 3.3.2).

- **Scalability**:
  Goal: The response time of the system must not degrade dramatically with the number of users and concurrent studies.

  The system is used by multiple teams with several users working on different studies. The work carried out by these users could be done concurrently and so the system will have a big workload when multiple requests are sent to the server. Since the users need study data quickly to conduct their research, it will be cumbersome if the users have to wait for a long time to get the data. Thus, the server should support quick data access for users. However, the system must do this in a way which takes memory into account to avoid an extensive resource consumption.

- **High performance**:
  Goal: to ensure a quick and responsive user interface by using a database, which facilitates fast data processing.

  To achieve this, we will implement an entity framework database, which utilizes Microsoft's .NET framework to optimize database queries. This takes advantage of optimized queries and features implemented by Microsoft database experts. Furthermore, a future feature may be implemented to cache the result of large queries for quick access. This allows the server to respond hastily to large queries, which have previously been used. This will only be implemented for queries on data that is less likely to change. As a result, a memory trade-off may occur, which can be negated by using limits on how much memory and how many queries are cached. By way of example, the last x number of query results to the server could be cached for repeated use. The design goal is a refinement of the non-functional requirement "high performance" in the Requirement Analysis Document (section 3.3.3)

## 2 Current Software Architecture

## 3 Proposed