

University College of Northern Denmark

Technology and Business

Computer Science Academy Profesion (AP) Degree

Class: DMAJ0916

Title: Technology and Programming Report – 3rd Semester 2017

Project participants (Group 2):

-Ralfs Zangis

-Andrei-Eugen Birta

-Hannes Heiskonen

-Stoycho Nenov Anastasov

Supervisors:

-Brian Hvarregaard

-Per Trosborg

Due date: 2017-December-18 (13:45 UTC+1)

Submission date: xxxx

Birta Zanigis Heiskonen Nenov

[1. Preliminary Study 3](#_Toc500259749)

[1.1. Introduction 3](#_Toc500259750)

[1.2. Project Idea 3](#_Toc500259751)

[1.3. Problem Statement 4](#_Toc500259752)

[2. Development Process 4](#_Toc500259753)

[2.1. Service Related 4](#_Toc500259754)

[2.1.1. Type of Service 4](#_Toc500259755)

[a. Choice 4](#_Toc500259756)

[b. Pros 4](#_Toc500259757)

[c. Cons 4](#_Toc500259758)

[d. Other possible choices 4](#_Toc500259759)

[2.1.2. Service’s Architecture 4](#_Toc500259760)

[a. Choice 4](#_Toc500259761)

[b. Pros 4](#_Toc500259762)

[c. Cons 4](#_Toc500259763)

[d. Domain Model 4](#_Toc500259764)

[e. Architecture Diagram 4](#_Toc500259765)

[2.1.3. Database Architecture 4](#_Toc500259766)

[2.2. Client Related 6](#_Toc500259767)

[2.2.1. Dedicated Client 6](#_Toc500259768)

[a. Choice of framework 6](#_Toc500259769)

[b. Pros 6](#_Toc500259770)

[c. Cons 6](#_Toc500259771)

[d. Other choices 6](#_Toc500259772)

[2.2.2. Web Client 6](#_Toc500259773)

[a. Choice of framework 6](#_Toc500259774)

[b. Pros 6](#_Toc500259775)

[c. Cons 6](#_Toc500259776)

[d. Other choices 6](#_Toc500259777)

[2.3. Middleware 6](#_Toc500259778)

[2.3.1. Client-Service 6](#_Toc500259779)

[a. Choice 6](#_Toc500259780)

[b. Pros 6](#_Toc500259781)

[c. Cons 6](#_Toc500259782)

[d. Other choices 6](#_Toc500259783)

[2.3.2. Service-Database 6](#_Toc500259784)

[a. Choice 6](#_Toc500259785)

[b. Pros 6](#_Toc500259786)

[c. Cons 6](#_Toc500259787)

[d. Other choices 6](#_Toc500259788)

[2.4. Security 6](#_Toc500259789)

[2.5. Concurrency 6](#_Toc500259790)

[2.6. Performance 6](#_Toc500259791)

[2.7. Interesting bits of code 7](#_Toc500259792)

[a. Keeping connection from timing out 7](#_Toc500259793)

[b. Callback methods in chat 7](#_Toc500259794)

[c. Database Triggers 7](#_Toc500259795)

[3. Conclusion 7](#_Toc500259796)

[3.1. Denouement 7](#_Toc500259797)

[3.2. References 8](#_Toc500259798)

# Preliminary Study

## Introduction

This document summarizes the collaboration of Group 2 for the programming and technology exam of the 3rd Semester. The group consists of 4 members of 4 different nationalities. Despite the major differences in our opinions (we agreed on certain rules and guidelines to follow, thoroughly elaborated in the accompanying document called group contract), we managed to harness the benefits of diverse ideas and identify multiple possible approaches to certain problems.

## Project Idea

Some of the problems we had to find the answer to include, but are not limited to: When a greater number of users, than the number of available places, wishes to enter a chat, at the same time, or a premade group, whose number of people is greater than the available slots in a given chatroom, tries to join (for example, a group of 4 attempts to join when there are only 3 or less slots left), preventing SQL Injection, learning how to use Callbacks and bindings.

After formulating the problem statement and having it approved by the supervisors we were assigned the task of solving the problem by selecting the most suitable agile system development method based on the situation, through well-planned and well-synchronized teamwork.

## Problem Statement

Some of the problems we had to find the answer to include but are not limited to: Users whose number is greater than the available slots in a given chatroom try to join at the same time (for example, there is 1 slot left but 2 people try to join), a group whose number of people is greater than the available slots in a given chatroom tries to join (for example, a group of 4 attempts to join when there are only 3 or less slots left), preventing SQL Injection, learning how to use callbacks and bindings.

After formulating the problem statement and having it approved by the supervisors we were assigned the task of solving the problem by creating a web service, selecting the appropriate middleware and create both a dedicated and a web client.

# Development Process

## Service Related

### Type of Service

#### Choice

#### Pros

#### Cons

#### Other possible choices

### Service’s Architecture

#### Choice

#### Pros

#### Cons

#### Domain Model

#### Architecture Diagram

### Database Architecture

A database is an organized collection of data. Databases can be designed to do just about anything with information – Track, organize, edit and produce reports. Many great applications and websites have a database as a main structural part.

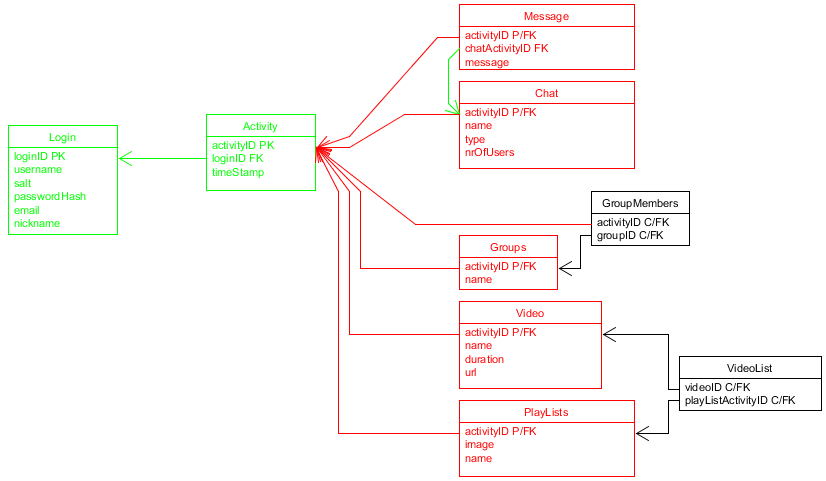
There are two types of database technologies: Relational Databases, which are great at organizing and retrieving structured data; and Non-Relational Databases, which are best used when the data is inconsistent, incomplete or simply massive.

For our project, we have chosen to go with a Relational Database, simply because pros such as: strict ACID support, data normalization, supports joins, limitless indexing, and being one of the most common used technologies\*, outweigh the of having a non-relational database, cons such as: working with joins can be difficult, low concurrency (no random reads and writes), slow mass updates, difficulty tracking schema changes.

As for engines, there are several choices that we considered, for a Relational Database, some of which are: Oracle Database, SQL Server and MySQL; and since all three of them were using dialects of the same language (SQL), it went down to the very basics when we took the decision on which to use.

As a final decision, we chose SQL Server 2014, because of the following: SQL Server executes and commits each instruction, unlike Oracle which requires explicit command to commit the changes; ease of use, since not only were we thought on how to use it, but also compared to Oracle, which give so many other settings and configurations that can be set to the wrong value; and performance.

The following diagram shows how our database currently looks.



As you can see, we have 3 types of tables: tables which hold critical information such as user details and actions (green), tables which specify the type of activity a user has done (red) and tables which’s main purpose is to handle a M-M relationship.

## 2.2. Client Related

### 2.2.1. Dedicated Client

#### a. Choice of framework

#### b. Pros

#### c. Cons

#### d. Other choices

### 2.2.2. Web Client

#### a. Choice of framework

#### b. Pros

#### c. Cons

#### d. Other choices

## 2.3. Middleware

### 2.3.1. Client-Service

#### a. Choice

#### b. Pros

#### c. Cons

#### d. Other choices

### 2.3.2. Service-Database

#### a. Choice

#### b. Pros

#### c. Cons

#### d. Other choices

## 2.4. Security

## 2.5. Concurrency

## 2.6. Performance

## 2.7. Interesting bits of code

### a. Keeping connection from timing out

### b. Callback methods in chat

### c. Database Triggers

Since one of the features, that would ease our programming task, we have decided in the begging was cascade deletion, and SQL Server 2014 does not allow such thing when a table is referenced, or references, by multiple foreign keys, we needed to find another way of dealing with phantom data. Shortly after the problem was encountered, we stumbled upon this ingenious way of doing it: Deletion Triggers.

What is a deletion trigger? Well, when a row is deleted from a table, instead of doing the default command, the server would execute this special piece of code, which specifies from which tables to delete and what.

The following image is one of the triggers present in our database.



As you can see, before deleting the entry that was commanded to be deleted, the engine would check in all of the mentioned tables if the entry’s foreign key, is referenced, then proceed to delete that entry, before continuing the check, and finally executing the commanded entry’s deletion.

# 3. Conclusion

## 3.1. Denouement

In conclusion, during this semester we have managed to achieve not only knowledge about different frameworks of developing software, a new programming language but also a new development concept, that being programming a distributed service.

As an ending note, we would like to thank all the readers, who invested their time in reading this paper and also to the guiding teachers that helped and guided us through the entire process.

## 3.2. References

\*According to James Serra, Big Data/Data Warehouse Evangelist at Microsoft in a presentation Published on Mar 15, 2016