This document summarises the collaboration of Group 2 for the system development exam of the 3rd Semester. The group consists of 4 members of 4 different nationalities. In spite of 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.

The idea of the project is to create a service that handles multiple chatrooms with limited people count capacity where people can discuss topics they are passionate about which the users may access either via the web client or the windows application. Listening to music through the application using YouTube Data API and joining groups are other major user stories of our program.

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 selecting the most suitable agile system development method based on the situation through well-planned and well-synchronised teamwork.

Part 2.1

We have decided to use a combination of the scrum and the XP methods of development taking daily meetings (usually at 10 a.m.) and sprint structure of the project work from scrum while ensuring the robustness of our program through XP and/or pair programming if necessary.

Pros:

* The daily meetings allow to very well understand the stage at which the project is as well as identify any possibly issues and setbacks as soon as possible before they become a major problem.
* Splitting the work into sprints provides the opportunity to evaluate how well-planned was each iteration and make changes if necessary (learn from our mistakes). For example, if we assign too many tasks and fail to accomplish them by the end of the sprint, for the next sprint we will be able to adjust and plan correctly.
* Quickly identify and solve any problems that were lurking and appeared not to be there.
* Pair programming can make different developers more familiar with the code overall rather than just their own part.
* Optimising and testing the code all the time greatly reduces the chance of having major bugs or missing functionality.

Cons:

* Planning meetings and travelling daily takes some time, which could’ve been used on work instead.
* The product owner can change their mind at any point.
* Pair programming can sometimes take more time than wanted since 2 people are focusing on the same task.