Assignment 1

Analysis and Design Document

Student: Eric Dumea

**Group: 30433**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Use JAVA/C# API to design and implement an application for a ping-pong association that

organizes tournaments on a regular basis. Every tournament has a name and exactly 8 players

(and thus 7 matches). A match is played best 3 of 5 games. For each game, the first player to

reach 11 points wins that game, however a game must be won by at least a two point margin.

The application should have two types of users: a regular user represented by the player and an

administrator user. Both kinds of uses have to provide an email and a password in order to access

the application.

# Functional Requirements

The functional requirements of this application are separated in terms of the user type:

* The regular user can perform the following operations:

- View Tournaments

- View Matches

- Update the score of their current game. (They may update the score only if they are one

of the two players in the game. The system detects when games and matches are won)

* The administrator user can perform the following operations:

- CRUD on player accounts

- CRUD on tournaments: He creates the tournament and enrolls the players manually.

# Non-functional Requirements

This being a desktop application for ping pong, it is assumed that a laptop or a computer is placed near the playing area, and the users are focused on playing, not taking account of the score. Consequently, the application features a user-friendly GUI with big buttons, not prone to accidental pressing. Also the menus are basic, both for the user and administrator.

- Performance: a response time of maximum 2 seconds

- Availability: 16hours/day

- Security: the passwords will be encrypted when sent to the database

- Testability: hardest part to test is the ending of the match/game and update score, this is done by both printing out results and Junit tests

- Usability: the application will work on any computer that has java installed

2. Use-Case Model

Use case: <use case goal>

Level: <one of: summary level, user-goal level, sub-function>

Primary actor: <a role name for the actor who initiates the use case>

Main success scenario: <the steps of the main success scenario from trigger to goal delivery>

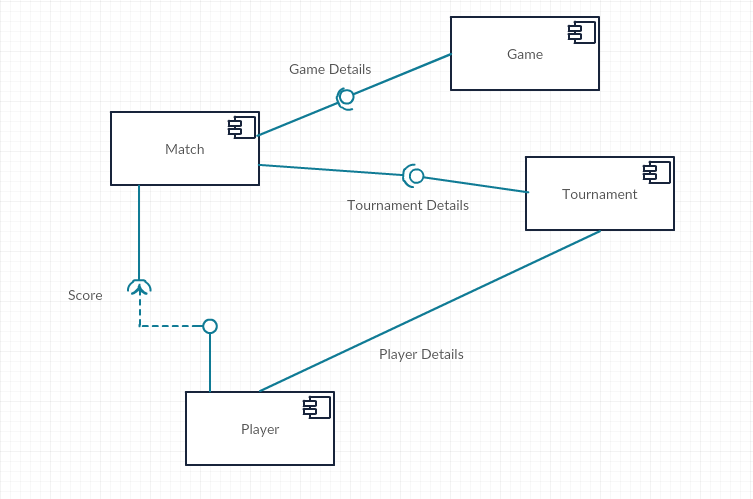
Extensions: <alternate scenarios of success or failure>

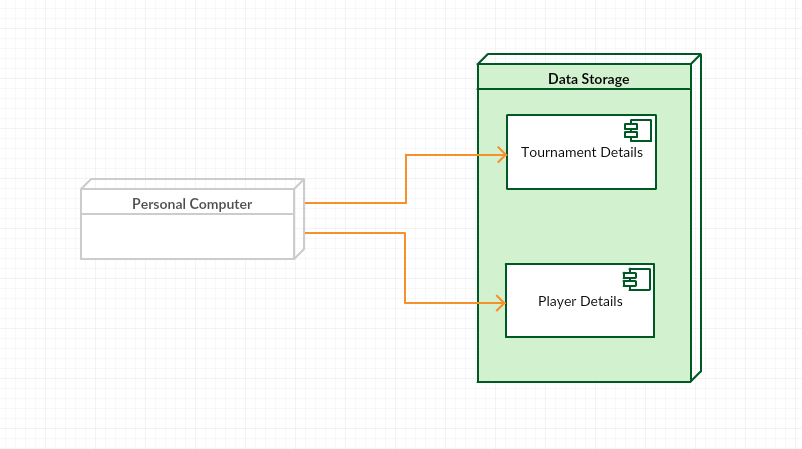
3. System Architectural Design

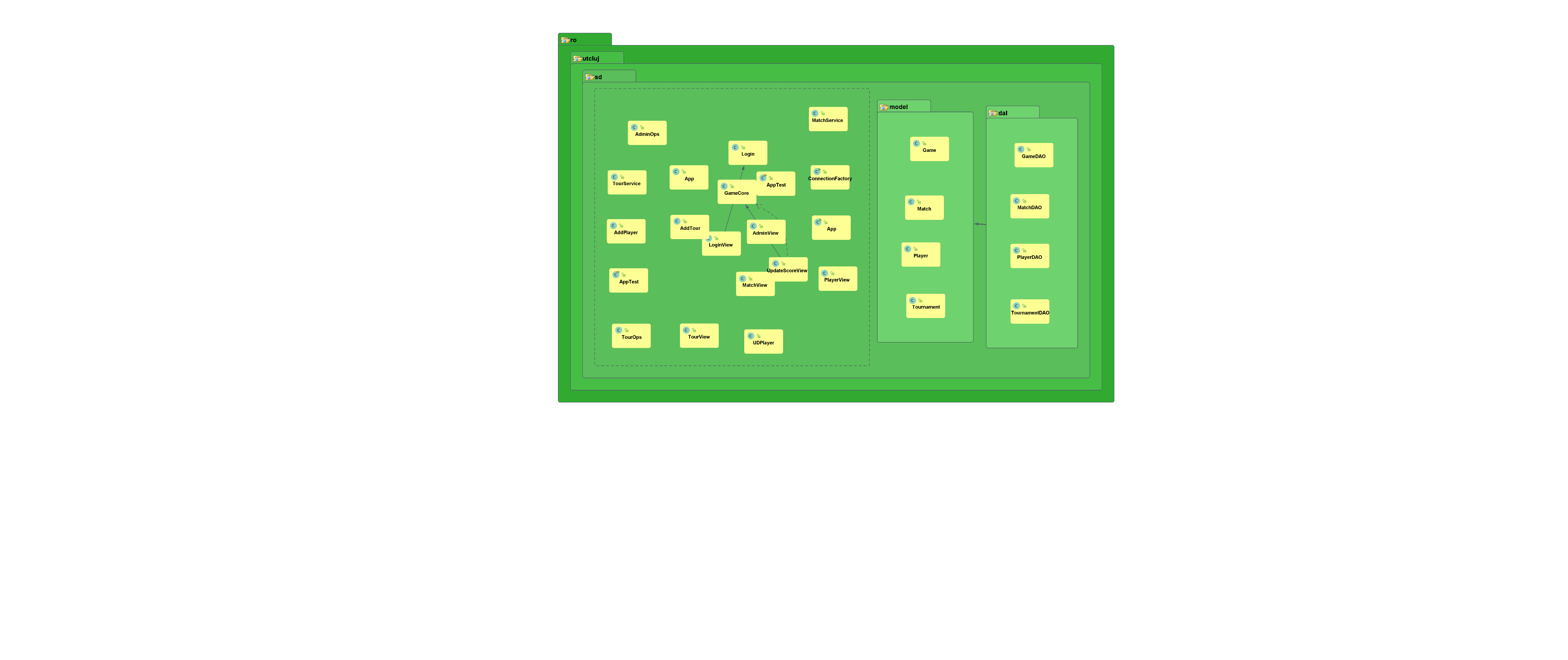
**3.1 Architectural Pattern Description**

In this project, as for architectural design patterns, the Layers architectural pattern was used. This pattern implies that there are physical “layers” of code. We used a 3-layer architecture, which was basically the Data Acquisition Layer, the Business Logic Layer and the Unit Interface Layer. We represented them in Java as being Maven modules, with dependencies between the DAO and the BLL and between the BLL and the UI.

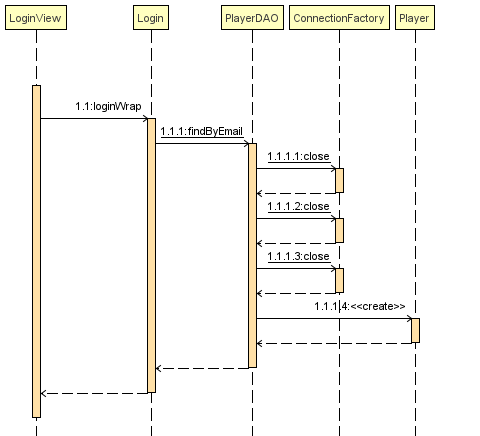
**3.2 Diagrams**



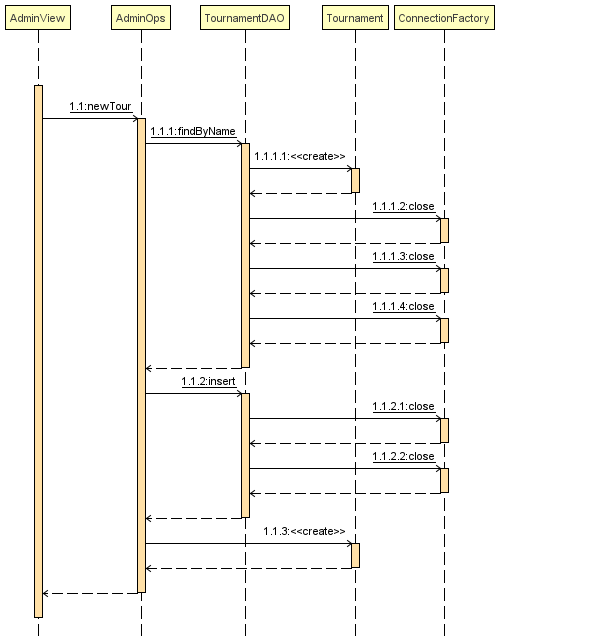




4. UML Sequence Diagrams

The UML Sequence Diagram I’ve found to be the most specific is the one for the logging in operation.

Another Sequence Diagram that I’ve found useful, and to be included, was the one in which an administrator adds a new Tournament.



5. Class Design

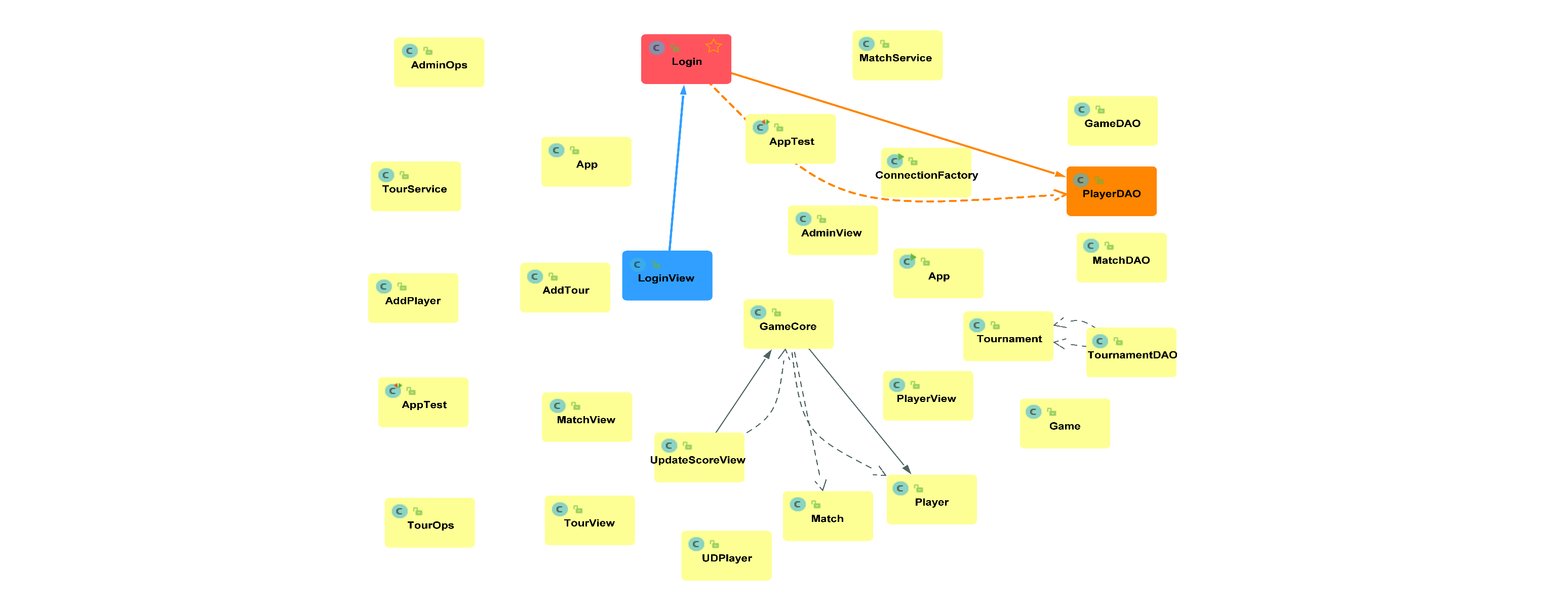
**5.1 Design Patterns Description**

Data Access Object Pattern or DAO pattern is used to separate low level data accessing operations from high level business services. Following are the participants in Data Access Object Pattern.

* **Data Access Object Interface** - This interface defines the standard operations to be performed on a model object(s).
* **Data Access Object concrete class** - This class implements above interface. This class is responsible to get data from a data source which can be database / xml or any other storage mechanism.
* **Model Object or Value Object** - This object is simple POJO containing get/set methods to store data retrieved using DAO class.

A Domain Model Design Pattern is one that creates a web of interconnected objects, where each object represents some meaningful individual, whether as large as a corporation or as small as a single line on an order form

**5.2 UML Class Diagram**

*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*

6. Data Model

The Data Models used in this application were the Player, Game, Match and Tournament, each having the corresponding Database fields and some specific additional ones for the ease of programming and maintaining the application.

7. System Testing

At the beginning the testing was done using printing functions (System.out.println()) to see that the data sent and received was correct. Later the testing was done using unit testing (JUnit) which involves testing parts (units) of the code to verify that they work as expected.

8. Bibliography

[1] - <https://github.com/buzea/SoftwareDesign2018/>

[2] - <https://www.youtube.com/playlist?list=PL6gx4Cwl9DGBzfXLWLSYVy8EbTdpGbUIG>

[3] - <https://pongworld.com/table-tennis-sport/rules>