Ping-pong tournament application

Student: Gyarmathy Tímea

**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

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.

# Functional Requirements

FR1: The application should have two types of users: a regular user represented by the player and an administrator user.

FR2: Both kinds of uses must provide an email and a password to access the application.

FR3: The regular user should be able to 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)

FR4: 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

* Availability: system must be available 98% per year.
* Performance: response time for any event should be less than 2 seconds.
* Security: only system administrators should be allowed to execute CRUD operations on database data.
* Testability: system must have a greater than 90% code coverage realized by unit tests.
* Usability: system must be clear and intuitive for types of users involved: ping-pong players, system administrators; any user should be able to log in 3 clicks;

NFR1: The data will be stored in a database.

NFR2: Use the Layers architectural pattern to organize your application. Use a domain logic pattern (transaction script or domain model) / a data source hybrid pattern (table module, active record) and a data source pure pattern (table data gateway, row data gateway, data mapper) most suitable for the application.

NFR3: All the inputs of the application will be validated against invalid data before submitting

the data and saving it in the database.

2. Use-Case Model

Use case: Login as Administrator

Level: User-goal level

Primary actor: System Administrator

Main success scenario:

1. User opens the app, the login window appears with possibilities to enter credentials
2. Administrator enters his e-mail
3. Administrator enters his password
4. Administrator clicks Log in button
5. System verifies credentials, sends request to database to verify if credentials exist
6. System verifies if user credentials have administrator privileges
7. System displays the Administrator Window with operations only available to the system administrator

Extensions:

In case administrator enters wrong credentials:

1. System displays a pop-up window stating, “Wrong credentials or user doesn’t exist!”
2. Return to step 2 (correct entered credentials)

In case user doesn’t have administrator privileges:

1. System opens Player Window with operations available to a registered player

Use case: Log in as Player

Level: User-goal level

Primary actor: Registered ping pong player

Main success scenario:

1. User opens the app, the login window appears with possibilities to enter credentials
2. Player enters his e-mail
3. Player enters his password
4. Player clicks Log in button
5. System verifies credentials, sends request to database to verify if credentials exist
6. System verifies if user credentials (don’t) have administrator privileges
7. System displays the Player Window with operations available to every player

Extensions:

In case player enters wrong credentials:

1. System displays a pop-up window stating, “Wrong credentials or user doesn’t exist!”
2. Return to step 2 (correct entered credentials)

In case user is not a player but an administrator, having system administrator privileges:

1. System opens Administrator Window with operations available only to system administrator

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>

*[Create the use-case diagrams and provide one use-case description (according to the format below).*

*Use-Case description format:*

*Use case: <use case goal>*

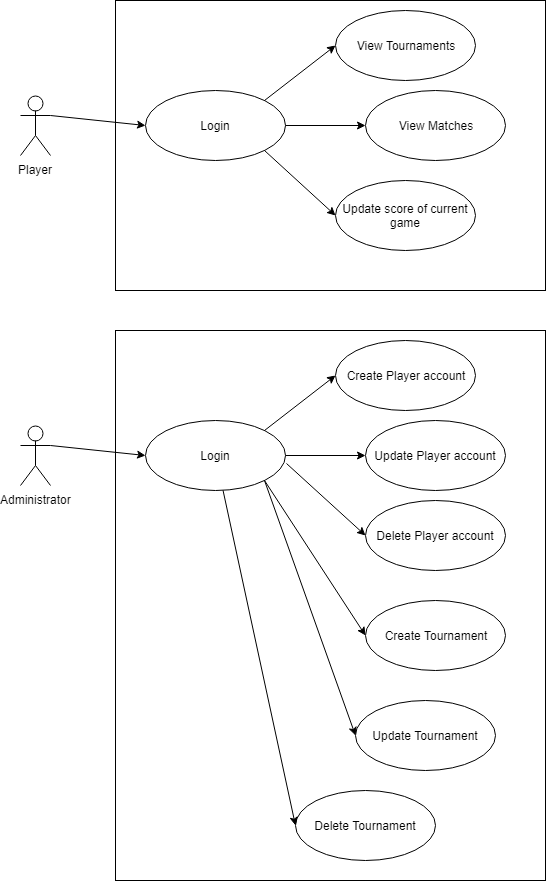
*Level: <one of: summary level, user-goal level, sub-function>*

*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**

*[Describe briefly the used architectural patterns.]*

**3.2 Diagrams**

*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*

4. UML Sequence Diagrams

*[Create a sequence diagram for a relevant scenario.]*

5. Class Design

**5.1 Design Patterns Description**

*[Describe briefly the used design patterns.]*

**5.2 UML Class Diagram**

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

6. Data Model

*[Present the data models used in the system’s implementation.]*

7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

8. Bibliography