TABLE OF CONTENTS

▸ Class-Diagram.md
· Domain-Model.md
· Installation-Manual.md
· Persistence.md
· Project-structure.md
· Requirements.md
· Source-Code.md
· System.md
· Testing.md
· Universal-design.md
· Usability-test-1.md
· Usability-test-2.md
· Usability-test-3.md
· Usability-tests.md
· Use-Case-Diagrams.md
· Use-cases.md
· User-Manual.md
· Wireframe.md
· home.md
· Sequence-Diagram.md

Class-Diagram.md

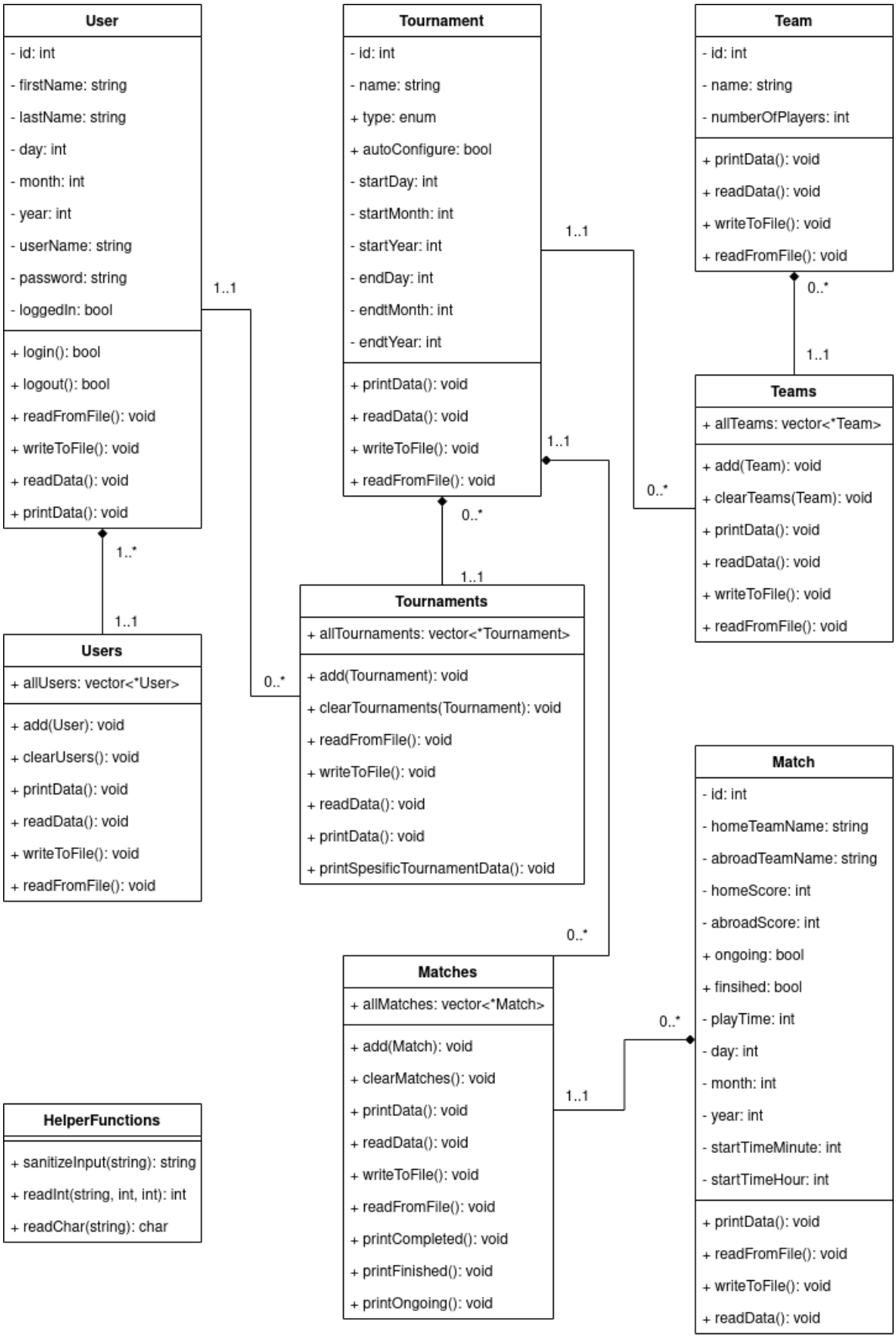


Figure 1 - Class Diagram

Domain-Model.md

Domain model

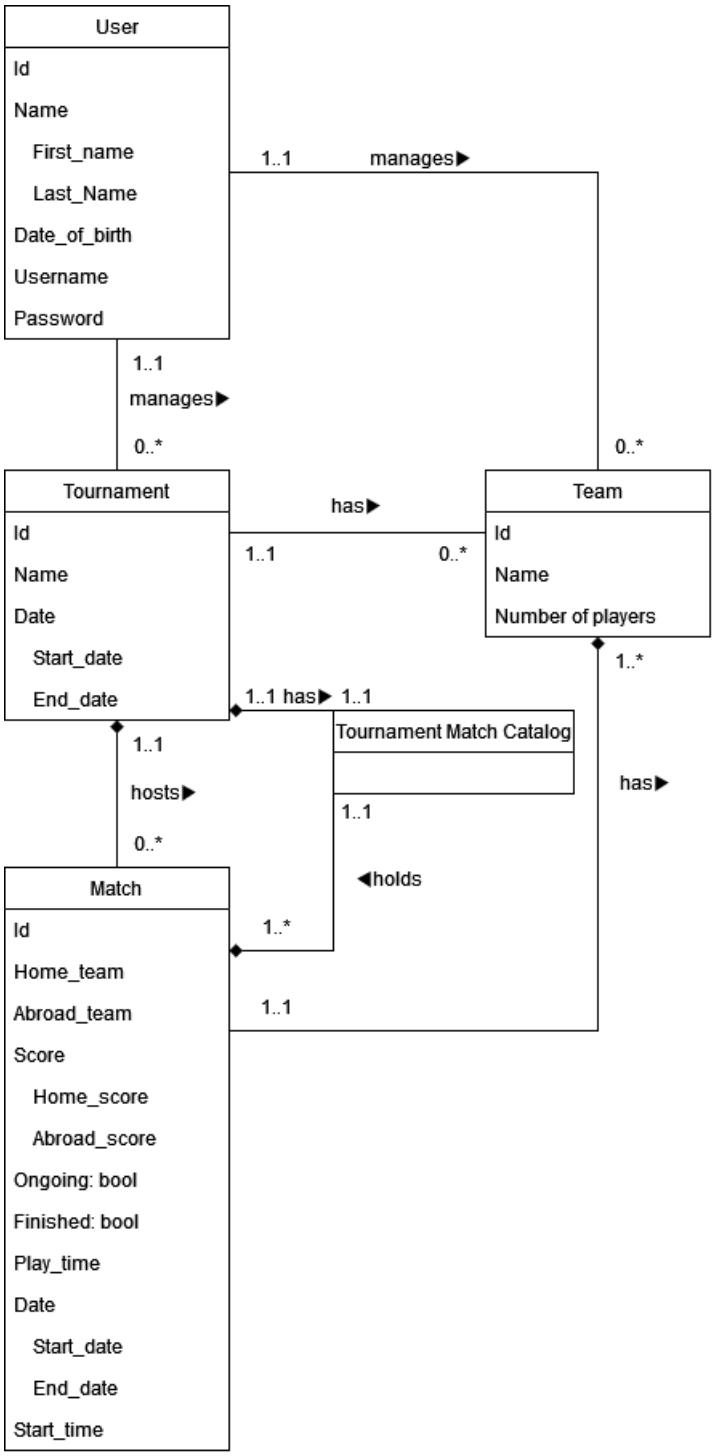


Figure 1 - Domain Model

Installation-Manual.md

User manual

Installing Maximus Cup Arranger

Precondition

You are using Microsoft Windows 10 as your operating system.

*Note that the program can be run on Linux, however we provide no official support for it, so if you want to use Linux you are on your own. The process of using the program on Linux is almost the same as it is for Microsoft Windows 10. Start by following the instructions in the **Second way** then compile the program and create a program file, then follow the instructions in the **First way**.*

There are primarily two ways of installing the application

The first way: running executable file

The first way which is also the easiest is to download the executable program file i.e. Maximus_Cup_Arranger.exe and run the file (Double left-click the file, or left-click and press the enter/return key). The executable file is located in the repository folder named **bin** *(folder path: PROG1004_2022_group3/bin)* or just use this link: https://gitlab.stud.idi.ntnu.no/sebasis/prog1004_2022_group3/-/blob/main/bin/Maximus_Cup_Arranger.exe . There are also three other important files in the bin folder which are: users.dta, tournaments.dta and teams.dta link to folder: https://gitlab.stud.idi.ntnu.no/sebasis/prog1004_2022_group3/-/tree/main/bin. These three files contain prefilled data and can be used for your convince when testing the program. You do NOT have to use these files if you do not want to, they will be autogenerated by the program when you start using it (the files will then only contain data you entered in the program).

The second way: compiling from source

Prerequisites: you know what and how to use CLI (command line interface), IDE (integrated development environment), Git (version control system for program code) and have experiance with the IDE CLion by Jetbrains or VSCode by Microsoft and CMAKE (https://cmake.org/).

1. Connect to the NTNU netwok, see VPN documentation here:https://i.ntnu.no/wiki/-/wiki/English/Install+vpn
2. Clone the repository to a directory of your choosing with this command: git clone https://gitlab.stud.idi.ntnu.no/sebasis/prog1004_2022_group3.git
3. Open the project in CLion or VSCode, see VSCode CMAKE documentation here https://github.com/microsoft/vscode-cmake-tools/blob/main/docs/README.md.
4. Build the project with CMAKE.
5. Run the program either through the IDEs*(console or debugger, default is shift+F10 or shift+F9 for debugger in Clion, F5 for VSCode.)* or run the program directly from the generated executable file.

NB!

Note that if you want to use the existing datafiles, they need to be placed at the same location as the executable program file.

Persistence.md

Persistence

The program's database consists of three .dta files which are:

- teams.dta
- tournaments.dta
- users.dta

The character '*' is used to separate the stored data.

Teams.dta

teams.dta holds all data which is relevant to the teams used in the application.
Example data and explanation for the different data members.

```
4
        // 4 indicates the total of stored teams.
0*Liverpool FC*LVP*11*
        // The first element is the team's unique id.
        // The second element is the team's name.
        // The third is the team name's abbreviation.
        // The fourth element is the total amount of players registered in the team.
1*Manchester City*MCC*11*
2*Bayern Munchen*BYM*11*
3*Real Madrid*RLM*11*
```

Tournaments.dta

tournaments.dta holds all data relvant to the created tournaments and their respective matches.
Example data and explanation for the different data members.

```
1
        // 1 indicates the total amount of stored tournaments.
T*0*VM*31*12*2100*31*12*2100*4*0*1*
        // T indicates that the data is of type Tournament.
        // The second element is the name of the tournament.
        // The next six elements is the start and end date for the tournament.
        // The tenth element is total amount of teams registered in the tournament.
        // The eleventh element is the tournament type.
        // The twelfth element is whether or not the tournament is autoconfigured.
2
        // 2 indicates the amount of registered matches.
3*Bayern Munchen*1*Liverpool FC*0*0*0*0*0*0*0*12*0*0*
        // The first element is the match's unique id.
        // The second element is the home team name.
        // The third element is the away team name.
        // The fourth element is the home team's score.
        // The fifth element is the away team's score.
        // The sixth element is the total playtime.
        // The three next values is the date (dd.mm.yyyy).
        // The eleventh is the start hour.
        // The twelfth is the start minute.
        // The thirteenth element is if the match is ongoing or not.
        // The fourteenth element is if the match is finished or not.
2*Manchester City*4*Real Madrid*0*0*0*0*0*0*0*21*1*0*
```

Users.dta

users.dta holds all relevant data for a user object.
Example data and explanation for the different data members.

```
2
        // 2 is the total number if users registered in the application.
0*sebastian*skylstad*14*1*1998*sebastian*okei*
        // The first element is the user's unique id.
        // The second and third elements is the first name and last name.
        // The three next elements is the date of birth.
        // the sixth element is the username of the account.
        // The seventh element is the account password.
1*Jan Olav*Aspelund*5*9*1999*Jan Olav*1234*
```

Project-structure.md

1 **bin** folder contains the application executable and data storage files for the application.
2 **build** folder contains cmake configuration and build files.
3 **docs** folder contains documentation made during the development process. Such as usability tests, minutes and meetings, wireframe, class and sequence diagrams etc.
4 **spike** folder contains MVP and a prototype.
5 **include** folder contains header files for the software.
6 **src** folder contains the source code for the software.

```
.
│   CMakeLists.txt
│   README.md
│
├── bin
│       .gitignore
│       Maximus_Cup_Arranger
│       Maximus_Cup_Arranger.exe
│       teams.dta
│       tournaments.dta
│       users.dta
│
├── build
│       .gitignore
│       CMakeCache.txt
│       cmake_install.cmake
│       compile_commands.json
│       CPackConfig.cmake
│       CPackSourceConfig.cmake
│       Makefile
│
├── docs
│   │   Gantt PDF.pdf
│   │   Gantt_diagram.png
│   │   Issue_Tracker_Week_1.png
│   │   Issue_Tracker_Week_2.png
│   │   Issue_Tracker_Week_3.png
│   │
│   ├── ClassDiagrams
│   │       classDiagram_v0.1.drawio
│   │       classDiagram_v0.1.png
│   │
│   ├── DomainModels
│   │       domainModel_v1.0.drawio
│   │       domainModel_v1.0.png
│   │
│   ├── Iterations
│   │   └── First Iteration
│   │           PROG1004_2022_group_3.pdf
│   │
│   ├── Minutes and meetings
│   │       01.03.22 Minutes.md
│   │       05.04.22 Minutes.md
│   │       08.03.22 Minutes.md
│   │       15.03.22 Minutes.md
│   │       19.04.22 Minutes.md
│   │       22.03.22 Minutes.md
│   │       28.02.22 Minutes.md
│   │       29.03.22 Minutes.md
│   │       User test 1.md
│   │
│   ├── SequenceDiagrams
│   │       sequenceDiagram_v0.1.png
│   │       sequenceDiagram_v0.1.svg
│   │       sequenceDiagram_v0.1_source.txt
│   │
│   ├── Universal Design
│   │       Universal Design.docx
│   │       Universal Design.md
│   │
│   ├── UsabilityTests
│   │       usability-test-1.md
│   │       usability-test-2.md
│   │
│   ├── Use-case diagrams
│   │       Use-case-1.png
│   │       Use-case-2.png
│   │       Use-case-3.png
│   │       Use-case-4.png
│   │
│   ├── Vision
│   │       Vision-document-rev-2.pdf
```

```
│   │
│   └── Wireframes
│           Wireframe-v2.png
│
├── include
│       .gitignore
│       constants.h
│       helperFunctions.h
│       match.h
│       matches.h
│       team.h
│       teams.h
│       tournament.h
│       tournaments.h
│       user.h
│       users.h
│
├── lib
│       .gitignore
│
├── spike
│   └── rapidPrototype_progDev_v0
│       │   LesData2.h
│       │   main.cpp
│       │
│       └── build
│               rapidPrototype_progDev_v0.exe
│
└── src
        helperFunctions.cpp
        main.cpp
        match.cpp
        matches.cpp
        team.cpp
        teams.cpp
        tournament.cpp
        tournaments.cpp
        user.cpp
        users.cpp
```

Requirements.md

- Use-Case diagrams
- Use-Cases (Text)
- Domain model
- Sequence diagrams
- Wireframe
- Universal design
- Usability tests

Source-Code.md

Source code

You must be on the NTNU network to get access to the repository, see VPN documentation here: https://i.ntnu.no/wiki/-/wiki/English/Install+vpn.

- The full repository is available here: https://gitlab.stud.idi.ntnu.no/sebasis/prog1004_2022_group3.
- The source files are located here: https://gitlab.stud.idi.ntnu.no/sebasis/prog1004_2022_group3/-/tree/main/src.

- The header files are located here: https://gitlab.stud.idi.ntnu.no/sebasis/prog1004_2022_group3/-/tree/main/include.

System.md

- Project structure
- Class diagram
- Persistence
- Source code
- Installation manual
- User manual
- Testing

Testing.md

Testing

Group number: 3 Command (module/function): Tournaments

Nr	Prerequisite	Action	Input	Expected output/result	Achieved result	Status (OK?)	Tester (name)	Comment
1	Picked Tournaments	Inputs command for tournament menu	2	Displays tournament menu	Displays tournament menu	ok	Sebastian, Andreas	
2	Pt.1	Select between the promoted tournaments	XXXX	Prints out all tournaments Selects a tournament	All tournaments printed out and selected a specific tournament	ok	Sebastian, Andreas	
3	Pt.2	Edit match, match id number	XXXX	Asks user for which match they want to edit. Loops until valid input.	Match selected	ok	Sebastian, Andreas	
4	Pt.3	Edit score	1	User is asked for which team to change the score in. Loops until valid input. Selects score between legal values.	Score changed.	ok	Sebastian, Andreas	
5	Pt.3	Change match status	2	Prompted to choose between setting a match to upcoming, ongoing, or finished loops until valid input is provided.	Status changed.	ok	Sebastian, Andreas	See Pt.6
6	Pt.3 match is set to finished			When X number of matches are finished the next matches (next brackets) are automatically added.	New matches/brackets added	ok	Sebastian, Andreas	
7	Pt.6	Set match start time	XXXX	User is a asked to set the starting time of the new matches created in Pt.6.	Start time set	ok	Sebastian, Andreas	
8	Pt.7	Continue to edit match	Y	Goes back to Pt.3	Loops back to Pt.3 and gives option to edit newly created match	ok	Sebastian, Andreas	

Figure 1 - Tournaments testing

Tournaments_testing.pdf

Group number: 3 Command (module/function): Users

Nr	Prerequisite	Action	Input	Expected output/result	Achieved result	Status (OK?)	Tester (name)	Comment
1	Program started	Create new user	2	User is asked to type in first name, last name, date of birth, username and password, loops until legal input is provided	User added to program.	ok	Sebastian, Andreas	
2	User exists	Login	1	User is asked to provide username and password. If username or password is incorrect, the user is asked if they want to try again. When login is successful a message is printed to console	User logged into their account ant a message is printed saying they are now logged into their account	ok	Sebastian, Andreas	The user now has access to the rest of the program features.

Figure 2 - Users testing

Users_testing.pdf

Group number: 3 Command (module/function): New tournament

Nr	Prerequisite	Action	Input	Expected output/result	Achieved result	Status (OK?)	Tester (name)	Comment
1	Add new tournament is picked (command 1)	Ask for name of tournament	XXXX	Name is cleaned for illegal characters and added.	Warning message if illegal characters were used, and name set.	ok	Sebastian, Andreas	
2	Pt.1	Assigns start date	XXXX	Legal date added	Start date set	ok	Sebastian, Andreas	Gives warning if using illegal input
3	Pt.2	Assigns end date		End date options later or equal to pt.1	Sets end date	ok	Sebastian, Andreas	Gives warning if using illegal input
4	Pt.3	Sets number of teams in tournament	XXXX	Total number of teams is set	Total number of teams is set	ok	Sebastian, Andreas	Allows only 2,4,8,16 teams in each tournament
5	Pt.4	Picks teams from list of teams	XXXX	Picks teams without duplicates and to the amount set in Pt.4.	Teams added to tournament with no duplicates	ok	Sebastian, Andreas	
6	Pt.5	Manual configuration	1	Starts manual configuration of tournament bracket (matches)	Starts config see Pt.7	ok	Sebastian, Andreas	
7	Pt.6	Picks two for each bracket	1 too Pt.4	Picks teams with no duplication.	Picks teams with no duplication.	ok	Sebastian, Andreas	Does Pt.8 and Pt.9 for all brackets.
8	Pt.7	Chooses home team	XXXX	Home team chosen, between the two teams configured in the bracket in Pt.7	Home team chosen	ok	Sebastian, Andreas	
9	Pt.8	Set start match time	XXXX	Input within 0 - 24 hours 0 – 59 minutes	Time set.	ok	Sebastian, Andreas	
10	Pt.5	Automatic configuration	2	Randomizes the brackets automatically	Brackets randomized	ok	Sebastian, Andreas	Does Pt.9 for all brackets.

Figure 3 - New Tournament testing

NewTournament_testing.pdf

Command (module/function): 2 (Teams)

Nr	Expectation	Action	Input	Expected reaction/ result (output)	Achieved result	Status (OK?)	Tester (name)	Comment
1	Command «2» is written	Menu for choosing to work with tournaments or teams is written.	2	No teams saved: Warning written before sent back to menu.	As expected	OK	Jan Olav Lyche Aspelund	
2	As in 1	As in 1	As in 1	At least one team in program: Teams saved in program printed before user is sent back to menu.	As expected	OK	Jan Olav Lyche Aspelund	

Figure 4 - Teams Show Teams Testing

Teams_showTeams_testing.pdf

Command (module/function): 1 (Add team)

Nr	Expectation	Action	Input	Expected reaction/ result (output)	Achieved result	Status (OK?)	Tester (name)	Comment
1	Command «2» is written	Start procedure to add new team.	Name for new team	Name already exists: "Team name already exists" written, then must re-enter name.	As expected	OK	Jan Olav Lyche Aspelund	
2	As in 1	As in 1	As in 1	Name is different from existing teams: continue team creation process.	As expected	OK	Jan Olav Lyche Aspelund	
3	2 is done	Asked to write team abbreviation.	Abbreviation for new team	Abbreviation not containing 3 characters: Promted to enter new abbreviation	As expected	OK	Jan Olav Lyche Aspelund	
4	As in 3	As in 3	As in 3	Abbreviation 3 characters long: Team creation process continued.	As expected	OK	Jan Olav Lyche Aspelund	
5	4 is done	Asked to write amount of team players.	Number of players (number)	Input is higher or lower than allowed limit or is not a number: Prompted to re-enter team size.	As expected	OK	Jan Olav Lyche Aspelund	
6	As in 5	As in 5	As in 5	Input is whole number within given parameter: Team creation process finished, and user sent to teams menu.	As expected	OK	Jan Olav Lyche Aspelund	

Figure 5 - Teams Add Teams Testing

Teams_addTeams_testing.pdf

Universal-design.md

Acronym - CLI (Command line interface)

WCAG 2.1

The application is expected to be designed according to theWCAG 2.1 principle 1 to achive high level of usability for any user, including those with disabilities.

Principle 1: Perceivable

This section of the guidelines is primarly focused making sure informastion is presented to the user in a way they can percive.

1.1 Text alternatives

The main goal is to have alternatives to non-text content by giving options such as braille, speech or symbols or simpler language. Since our application is only accessible through a command line interface this point is non-applicable.

1.2 Time-based Media

Providing alternatives for time-based media such as audio clips or videos by giving alternatives like cations or sign language. This point is non-applicable since our program only involves text.

1.3 Adaptable

Main goal is to create content that can be offered in multiple ways without losing structure or information.

1.4 Distinguishable

Make it easier for users to see and hear content including separating foreground from background.

Success Criterion 1.3.4 Orientation There are limited options available for customization on a CLI, but one way we can impact the user experience is text wrapping. By making sure the text doesn't go out of the window size we avoid confusion and make it easier for the user to understand what is being presented on the screen.

Success Criterion 1.4.1 Use of Color The command line interface will only display text in the terminal. The colour of the background and text is not something we can control, but is based on the colour scheme of the users terminal. This is usually a plain black or blue background and plain white or green text colour.

Success Criterion 1.4.4 Resize text The command line interface allows users to resize their text without the loss of content or functionality. Either by manually changing the setting of zoom, or by using shortcuts like Ctrl and +/- on Windows or Command and +/- on Mac OS.

Success Criterion 1.4.10 Reflow The program does not require users to scroll when presenting information. New information displayed will always emerge bottom up on the CLI. Whenever new text appears the CLI will always scroll down to the button where the new information is displayed.

Don Norman's Principles of interaction design

It is also expected that the application relates to Donald Norman's principles of interaction design to ensure high usability and good user experience.

Visibility: Our program is made to be used only in text format in a command line interface. Since we are making the program, the information displayed for the user will all be relevant. Available options are described on screen in a simple understandable way. Information regarding how to navigate the program is written in more detail in the user manual.

Feedback: After every action taken by the user the program will give feedback indicating if the entered data is accepted/declined or if something went wrong. Options users have to go forward will also be presented after each action is taken. Since the program is only available in a command line interface, the history is always easily accessible for the user by scrolling up.

Constraints: Since we are making the program ourselves, we have full control over which actions the user can take at any given place in the program. The program will constrain the user by only allowing input that we have predetermined as acceptable.

Mapping: A command is linked to each option in the different menus. The user will be shown which options are available at any given moment and easy keyboard presses performs each option.

Consistency: For the sake of simplicity and improving the user experience similar actions have the same input through the program.

Affordance: Since our program is only available in a command line interface, the prior knowledge a user has with a CLI will determine how easily they pick up on the program. The user manual will help users understand how to use the program, but the program will also display all options available at any given moment.

Usability-test-1.md

Introduction

Maximus Cup Arranger aims to make the process around adding, configuring and updating football cup tournaments easy. It serves as a football cup tournament manager in which a user can easily configure their cup tournaments.

Summary

The main takeaway from the user test were that the program for the most part was intuitive to use, but it lacked clarity in some areas which made the user’s expectations differ from what a program command did and what happened e.g., when the commands ‘T’ then ‘A’ was used to get to add teams in a tournament, and the page with ongoing and completed matches were unclear.

People Involved

There was a total of four participants:

1. An interviewer.

2. A person who took notes.
3. An observer.
4. The user.

Demographic

The sample size of the user test is limited to one user due to time constraints and difficulties in finding multiple participants which had high amount of experience in the same tournament systems. The age group of the demographic is late 50’s with a decent amount of experience in computer and application use. Our expectations were that the user would most likely find some functionality unintuitive which we as system designers might consider “self-explanatory”.

Tasks

The user had to complete predetermined sequences of interactions with the program in the form of commands, numerical and string inputs to explore the application. This ensured that as many aspects as possible would be explored. See 17.1.1 Interview questions and steps to go through with user for the first user testing session 02.03.2022. in the appendix.

Results

Most features are good, and the program is for the most part intuitive to use. Some aspects the user displayed their concerns about were:

- How the adding teams to a tournament were unclear and confusing
- The ongoing and completed match log provided poor readability.
- Confusion regarding tournament types.
- When adding team or tournament the user expected their respective results to be printed automatically.

Usability-test-2.md

Introduction

Maximus Cup Arranger aim to make the process around adding, configuring, and updating football cup tournaments easy. It serves as a football cup tournament manager in which a user can easily configure their cup tournaments. There were three participants, For the second user test, we have let four users test out our first minimum viable product (MVP).

People involved

There were three participants present, an interviewer, an observer, and the user. The interviewer was responsible for interacting with the user, asking questions, and answering the users’ questions. The observer was responsible for noting the users´ comments, first impressions, navigational choices, satisfaction rating and issues they may face while using the program.

Summary

The main takeaway for the usability test with the MVP was that it was intuitive to use, but with some clear functionality shortcomings. The usability testing helped us figure out what would be most important to the user. E.g., adding scores to a tournament.

Demographic

We wanted to include a wider demographic in this usability test but due to time constraints and difficulty in finding participants in a wide age-group, we ended performing user tests on the following groups: Three in their early 20s, and one in their early 50s. Three of the users had experience with CLI-based applications, while one user had little-to-no experience with non-GUI applications. A total of three out of four participants had an interest in football, while one of the users had little-to-no interest in the sport.

Tasks

Since we now had our MVP mostly functioning, we could test most of the application´s features, even though aspects like adding scores to a team has yet to be implemented.

Results

Usability problems

The main complaint among all testers was the lack of ability to edit the tournaments. In the programs current state, the program is not able to perform this function. This issue is going to be resolved in the final product. Another concern was the program prompts, which some users found unclear. E.g., one user was spooked by the warning “Cannot find user.dta file”. There were also questions from most users regarding the option for choosing “Home and away”. This is going to be refined in the final product. One user experienced an issue with the program entering a loop after inputting the same team’s name twice, as well as when the user chose the same team against itself in the tournament creation process. This is also going to be fixed in the final version of the product.

Other feedback

A lot of the feedback received was directly based on the usability of the application. Elements such as being able to go back to the main menu after going to tournaments, as well as getting feedback about entered values were wishes from most users. The possibility to abort a process was also mentioned.

Conclusion

The users provided valuable feedback for the continuation of the development of the application. Based on the information provided, we are going to fix the following issues:

- More spacious text in certain parts of the program
- Print out tournament information after creation.
- Ability to import existing teams to a match.
- Being able to go back to the main menu after entering tournaments
- Change the format name other to manual
- Be able to abort a process
- Hide password field
- Ask for a password confirmation under user creation
- Text to show which team is currently added, instead of the current “Write name for new team”, which does not show which teams that are available
- Ability to remove teams already assigned
- Fix randomize-option

Usability-test-3.md

Introduction

This is the third and final user test report based on tests on users. This test was done to make sure our software (Maximus Tournament Arranger) was in a good enough state to be released.

Summary

Compared to the last tests, this time there were generally only small issues we got negative feedback on, and even then it was in an optimistic way. This time the program was more polished, and most of the test went almost entirely pain-free. There was one major bug that appeared as well as a few typos here and there, but overall the

test went really well.

People involved

3 people:

- One interviewer
- One taking notes (and moght come with separate questions)
- One test participant

Demographic

The software is targeted towards people from 30-50 years old, who needs a program to keep track of one or multiple simple football tournaments. The usability test is to find any unintuitive spots of the software and give us pinpointers towards making the software more intuitive for new users.

Tasks

The tester was asked to do tasks like logging in, create and view the created team, and create and edit a tournament. These tasks are done to see how intuitive and understandable everything is. During the test the tester was asked to describe if anything felt intuitive or not, and whether some things could be improved on.

Results

Most tasks went without any issues, with the exception of these:

- The automatic creation of new matches using the winners of previous matches was not obvious at first glance, and there were no messages stating a new match was being created other than a list of two teams and a choice of which of them to make the home team.
- A bug happened where the both teams from one of the last matches was put into the new match, when only the winner from that specific match should go on.
- The bracket system was not obvious until a tournament with 8 teams was created and tested.

A few typos were discovered during the test as well.

Conclusion

As most of the test went well and not many complaints were made, it was decided that the software only needed some bug fixes and minor changes before the final deadline. The program itself worked well, and was more or less polished enough to be released soon.

Usability-tests.md

Usability test 1 \ Usability test 2 (MVP) \ Usability test 3 (finished product)

Use-Case-Diagrams.md

Use-Cases (Text)

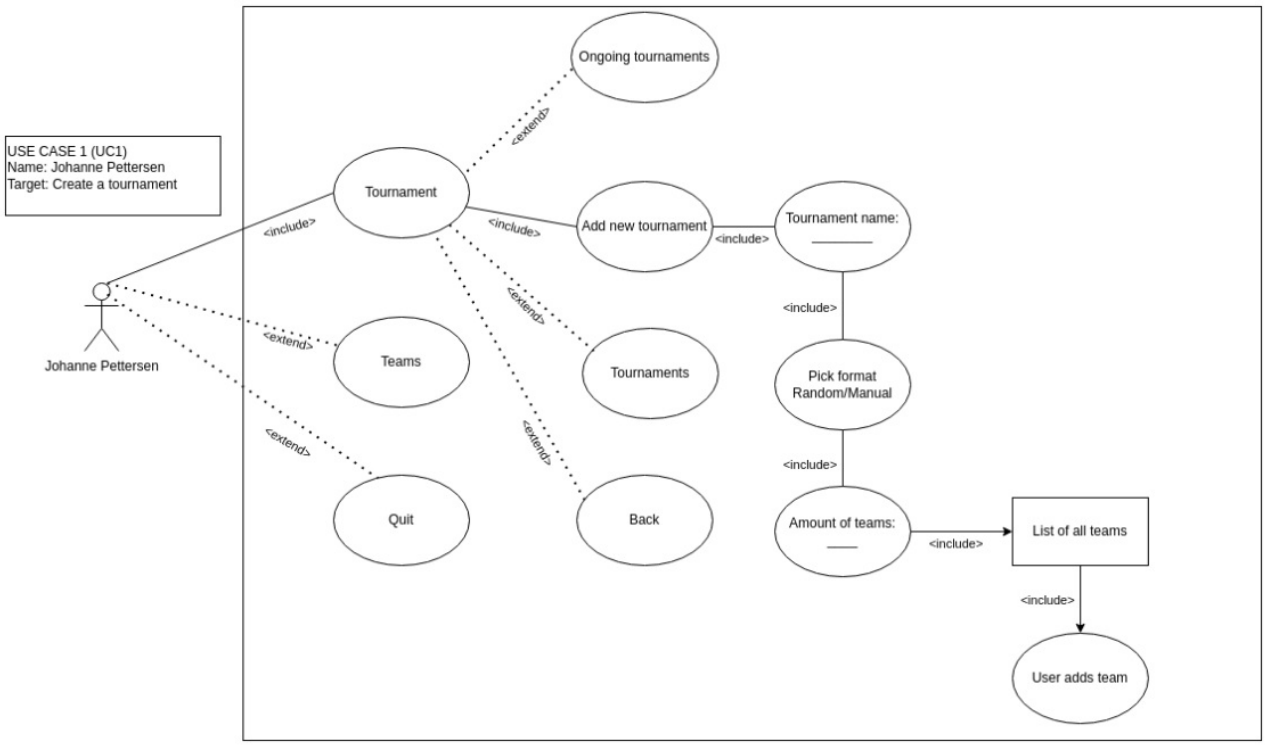


Figure 1: Use case 1

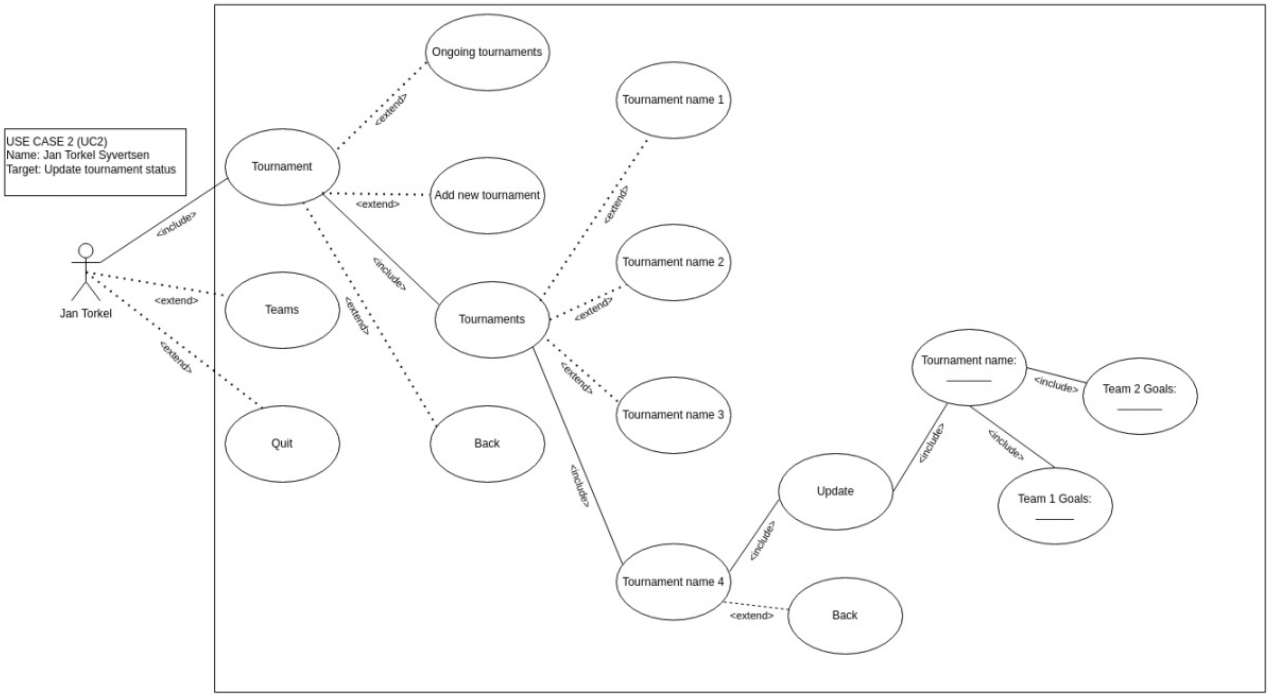


Figure 2: Use case 2

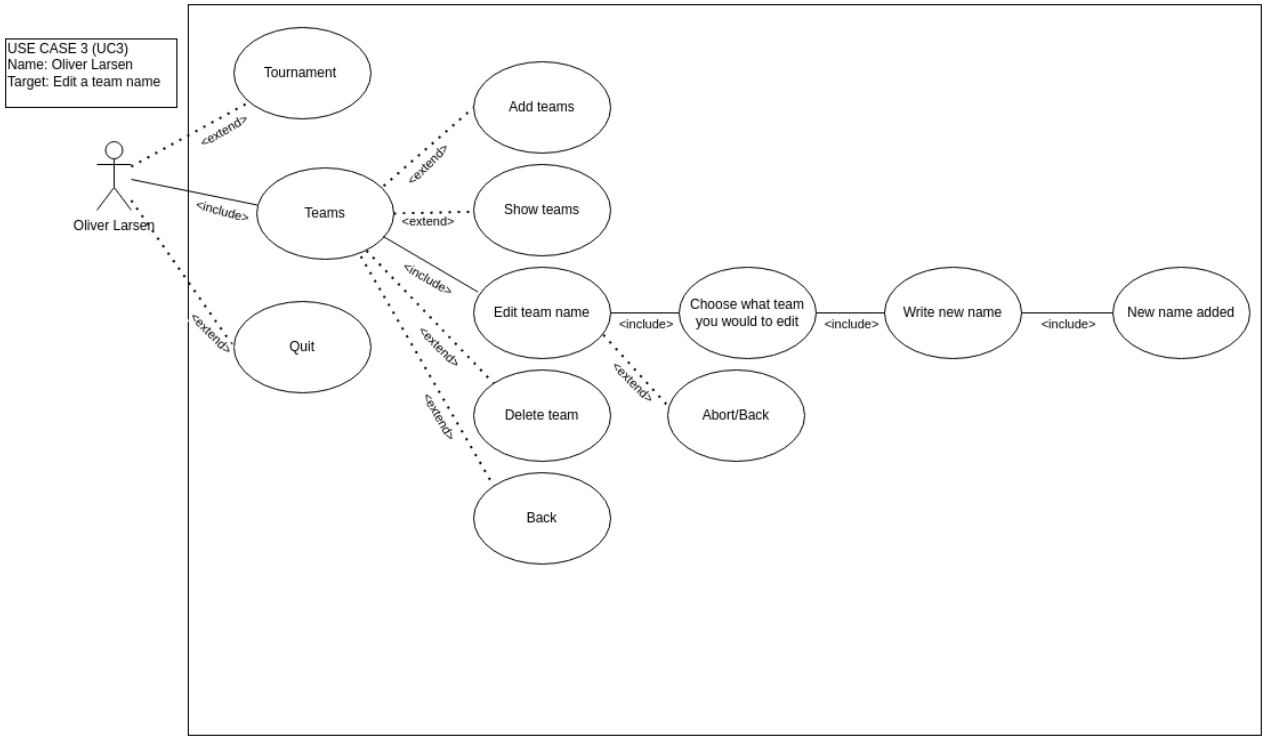


Figure 3: Use case 3

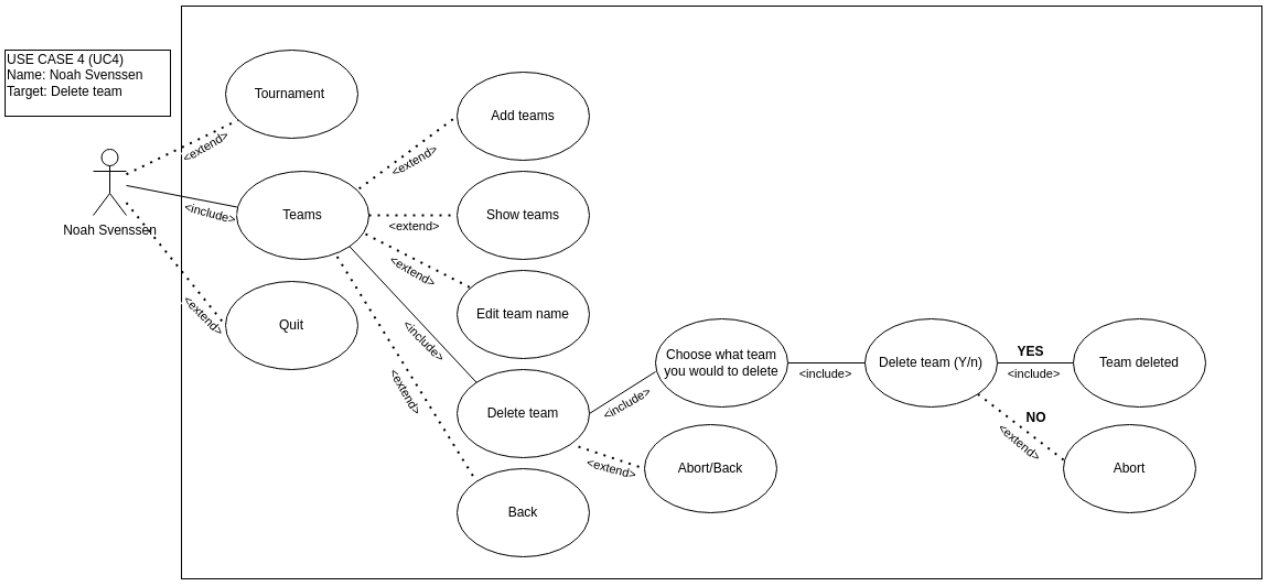


Figure 4: Use case 4

Use-cases.md

Use case 1

Info	Description
Goal	Create a tournament
Trigger	User wants to create a new Tournament
Pre-condition	None

Info	Description
Main cycle	1. User wants to create a tournament 2. User clicks on “Tournament” 3. User clicks on “Tournaments” 4. User clicks on “Add new tournament” 5. User adds Tournament Name 6. User clicks on chosen format Random/Manual 7. User types how many teams he wants in the tournament 8. User selects 4 different teams based on the integer in front of team name 9. The tournament is now made and saved on the program.
Successful action	The user is shown all relevant information about tournament format

Use case 2

Info	Description
Goal	Update tournament status
Trigger	User wants to add goals to teams after a match
Pre-condition	Must already be tournaments existing on the program
Main cycle	1. User wants to check tournament format 2. User clicks on “**Tournament**” 3. User clicks on “Tournaments” 4. User chooses one of the tournaments presented 5. User clicks on “Update” 6. User types in goals for each team.
Successful action	The user has successfully updated the scores of a match.

Use case 3

Info	Description
Goal	Edit team name
Trigger	User wants to edit a team.
Pre-condition	Teams must exist in the program.
Main cycle	1. User wants to edit a team 2. User clicks on “Teams” 3. User clicks on “Edit team name” 4. User chooses which team he/she would like edited 5. User enters new team name 6. The new team's name is now saved in the program
Successful action	Team has successfully been updated.

Use case 4

Info	Description
Goal	Delete team
Trigger	User wants to delete a team.

Info	Description
Pre-condition	Teams must exist in the program.
Main cycle	1. User wants to delete a team User clicks on “Delete team” 3. User chooses which team he/she would like deleted. 4. User confirms deletion of team. 5. The team is now deleted.
Successful action	Team has successfully been updated.

User-Manual.md

Installation manual

User manual info

Audience

This document is intended for individuals using Maximus Cup Arranger to manage football tournaments.

System requirements

- Microsoft Windows 8 or later
- 20 MB of free space

Overview of the Software

Maximus Cup Arranger

Maximus Cup Arranger is an application made for managing football tournaments. This application will keep data about any tournaments with up to 16 teams at once. As it is entirely client based and requires next to no computational power, the application will never run slow. Only keyboard input is required to use this software.

Using Maximus Cup Arranger

General Information

Maximus Cup Arranger is a CLI application, CLI stands for Command line interface. There are no interactive pictures or graphics, it’s a text-based program. This means that the only way to interact with the application is with text.

Creating an account

Creating an account is the first step to start using Maximus Cup Arranger. When you start the application, you will first enter the User Menu. In the user menu there are three options available.

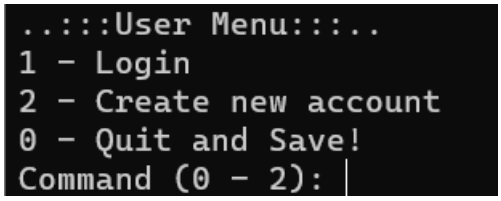


Figure 1 – UI of user menu

In this scenario the only valid input is numbers in interval 0-2 Pick the second option to create an account by entering the number “2” and pressing enter. You will then be prompted to enter a First name, Last name, Date of birth, username, and password.

Example of user creating an account.

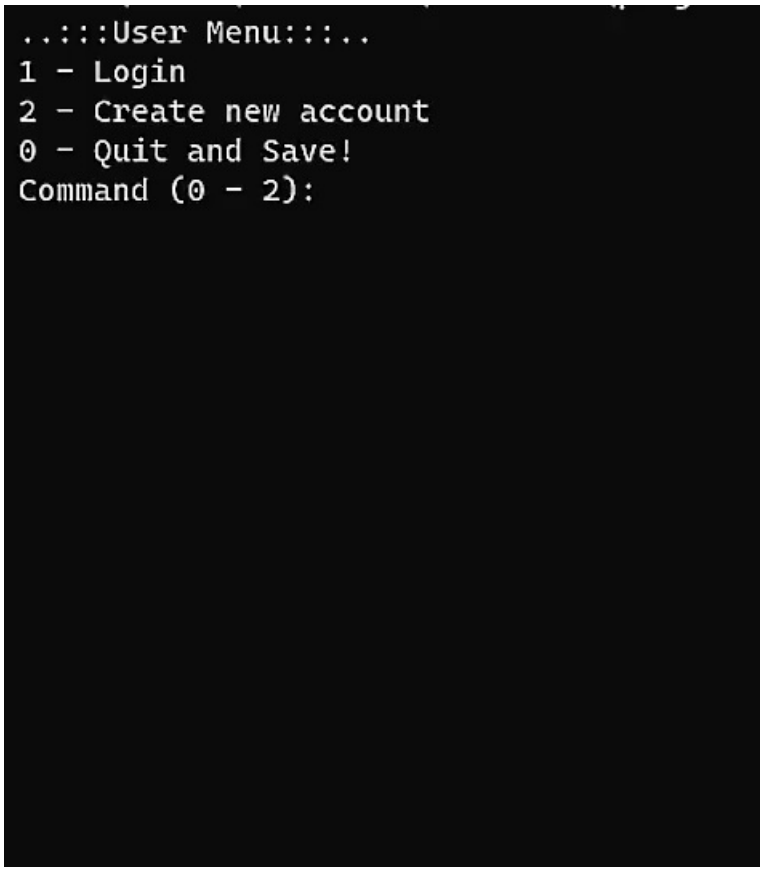


Figure 2 – UI from user creation.

Creating a team

After you have created an account, you can log in. You are then directed to the main menu.

Figure 3 – UI of logging in.

To create a team you must enter the “teams menu” by entering “2”. Enter “1” to start making a new team. Then simply enter the name for the team, an abbreviation for the name (only 3 characters long) and amount of players on the team. Note: these 3 are all entered separately.
Team creation Example

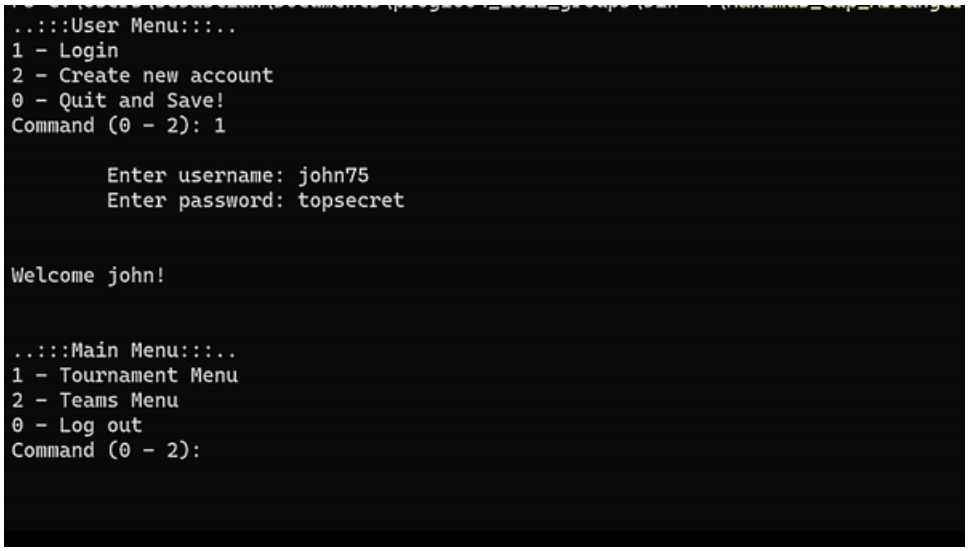


Figure 4 – UI of teams menu.

From teams menu press “2” to view all teams.

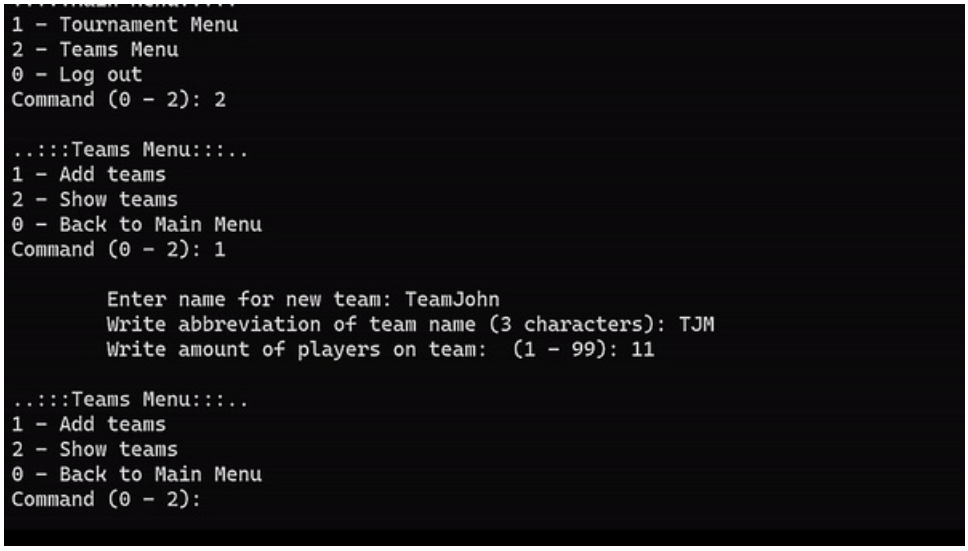


Figure 5 – UI of all teams.

Creating a tournament

To create a tournament, you need to be logged in. Then enter “1” to enter the tournaments menu, and then enter “1” again to start setting up a new tournament.

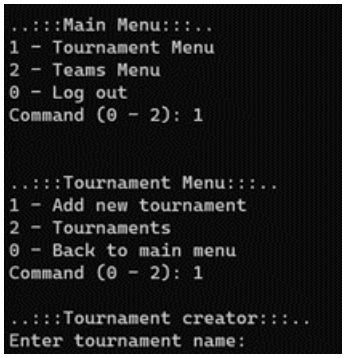


Figure 6 – UI from entering the tournament creator.

Start by entering a name for the tournament. Then enter start year, start month, and start day. Note: year, month and day must be entered separately. Same for end year, month and day. Then you have to choose how many teams there will be in the tournament, up to a maximum of 16. Only 2, 4, 8 and 16 teams can be entered, and they must all have been added through the teams creation process (segment above). Then choose the teams that will be in the tournament from the list that is written by entering their numbers one at a time. Once the required teams have been entered, you must choose between manual and random match setup.

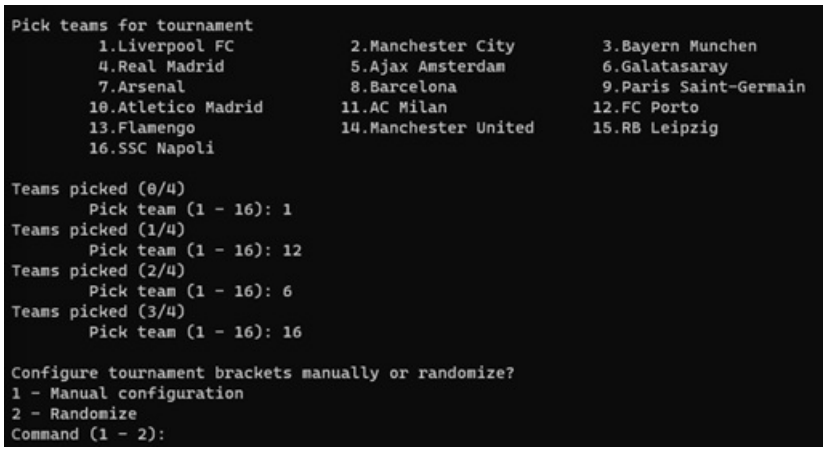


Figure 7 – UI of team selection for tournament creation.

Random will put teams against each other randomly while manual will make you choose which teams go against which. If you choose manual: simply pick which 2 teams from the list of selected teams you will put against each other. Both choices will then prompt you to choose which team is on homefield, and then ask for the time the match is supposed to start. This will be done for each match. Once all matches have been configured, you will return back to the “tournaments menu”. Congrats! A tournament has just been set up!
Example tournament setup with manual configuration

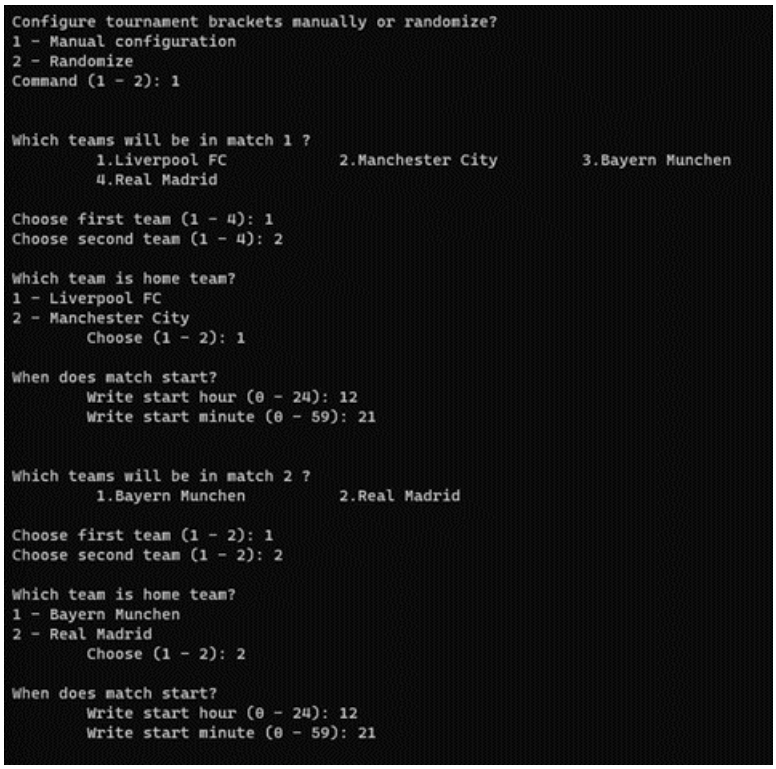


Figure 8 – UI of updating a tournament.

Updating a tournament

To access tournament editing you need to be in the “tournaments menu” (after logging in, enter “1”). Then enter “2” to show a list of all tournaments saved. Choose the one you want to edit by entering the number for it. The main info for the tournament including all matches will be printed on screen, and you get the choice if you want to edit a match. Enter “y” (capital or lowercase) and then choose the match you want to edit.

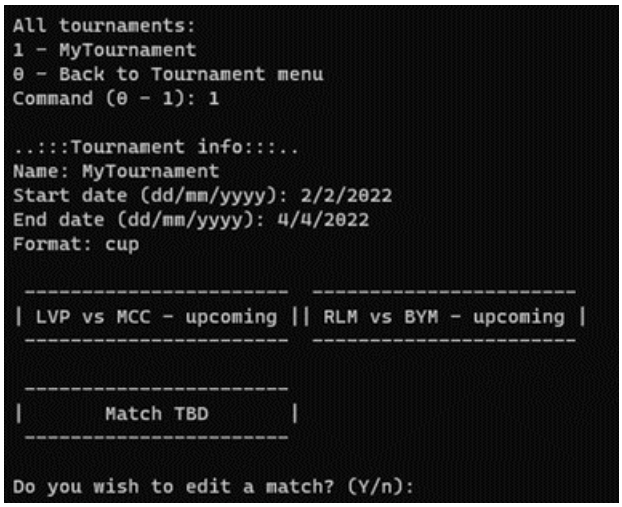


Figure 9 – UI of tournament viewing screen.

From there, you can enter “1” to choose to change the score of one of the teams. Enter the number of the team you want to change score on and enter the new score. The other command (2) will allow you to change the match status (upcoming/ongoing/finished). Enter the number associated with the status. Finally, you get the choice to continue editing or exit to the “tournaments menu”. Once all current matches have been set to “finished”, the winners of each match (the team with the highest score) will be automatically placed against each other in the next brackets. Starting time and home team must be entered for each match though. Once all edits have been done, you will be brought back to the “tournaments menu”.

Saving data

To make sure your data is securely saved, exit the software by logging out before closing the application. If not, everything done since you started the software might not be saved. Logging out is done by entering “0” in the main menu. Then you can close the program by entering “0” again or simply press the x-button in the top right corner of the window.

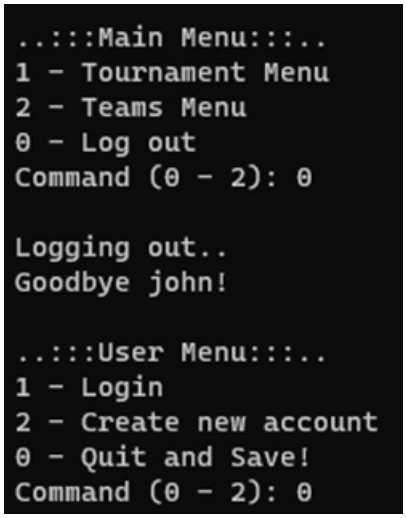


Figure 10 – UI of logging out.

Troubleshooting

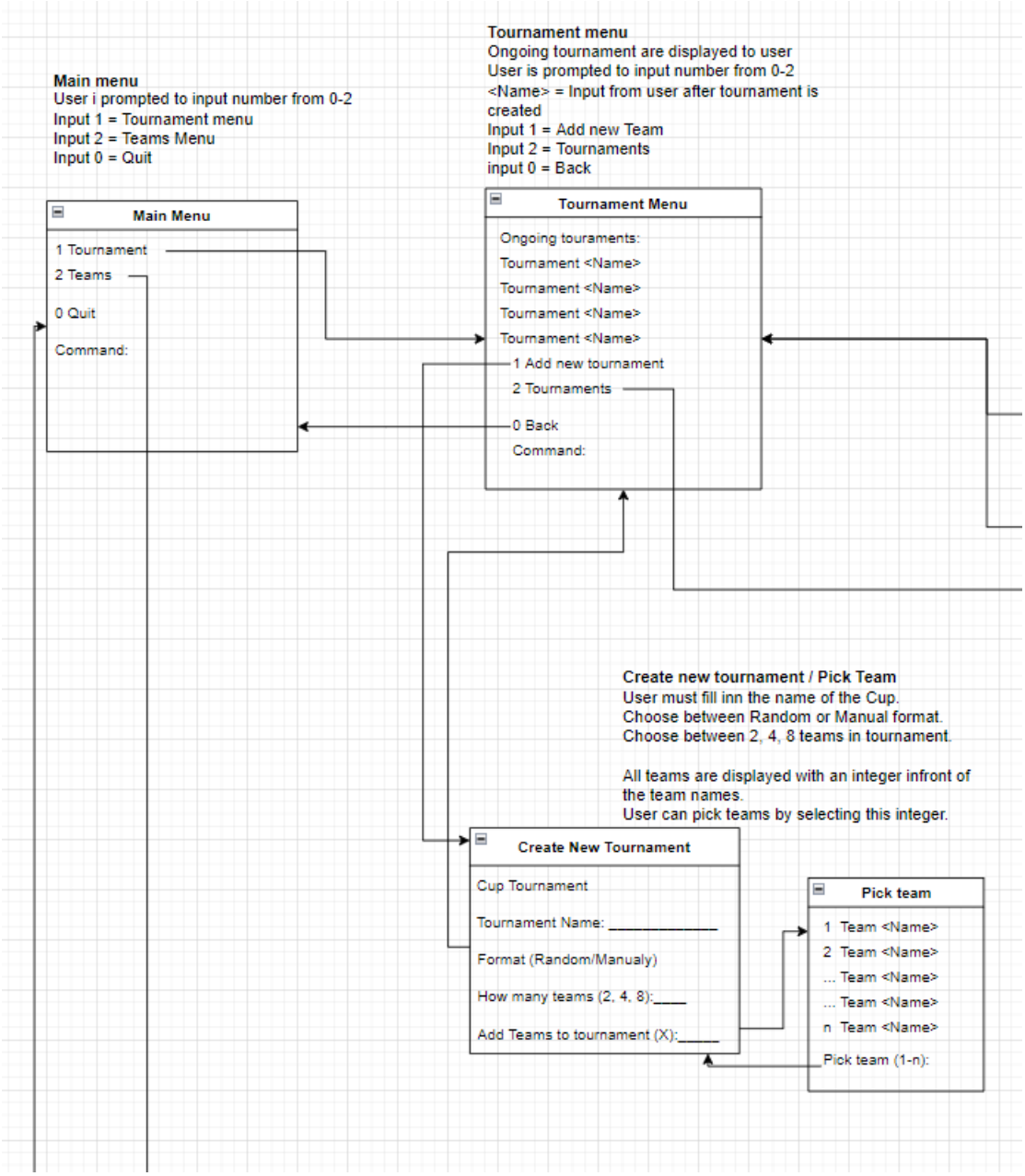
Common Problems

Some crashes can be fixed by deleting users.dta, teams.dta and tournament.dta. This will delete all your saved data.
To prevent this from happening, allways exit the program by pressing 0,

Support

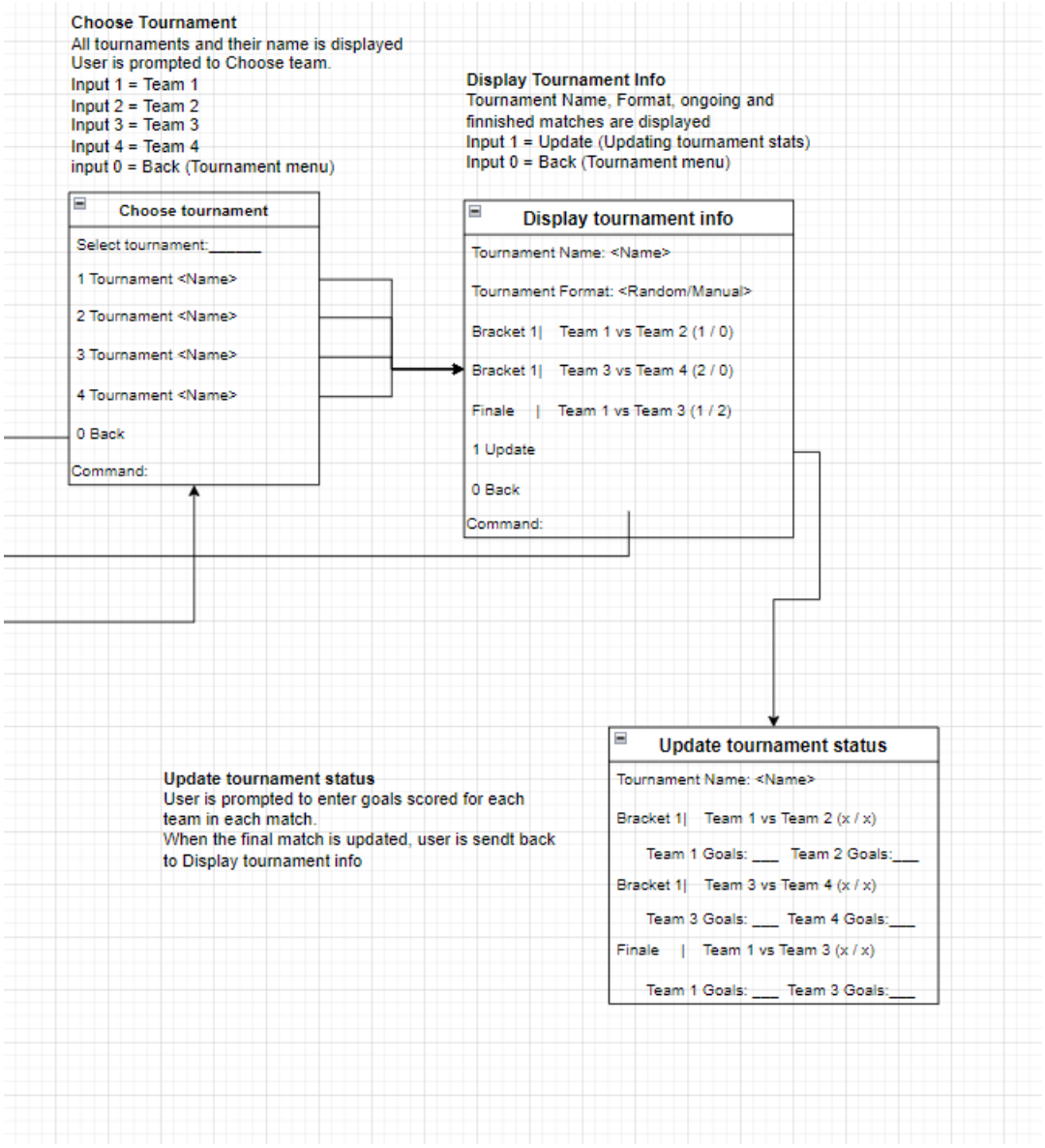
For support contact sebasis@stud.ntnu.no.

Wireframe.md

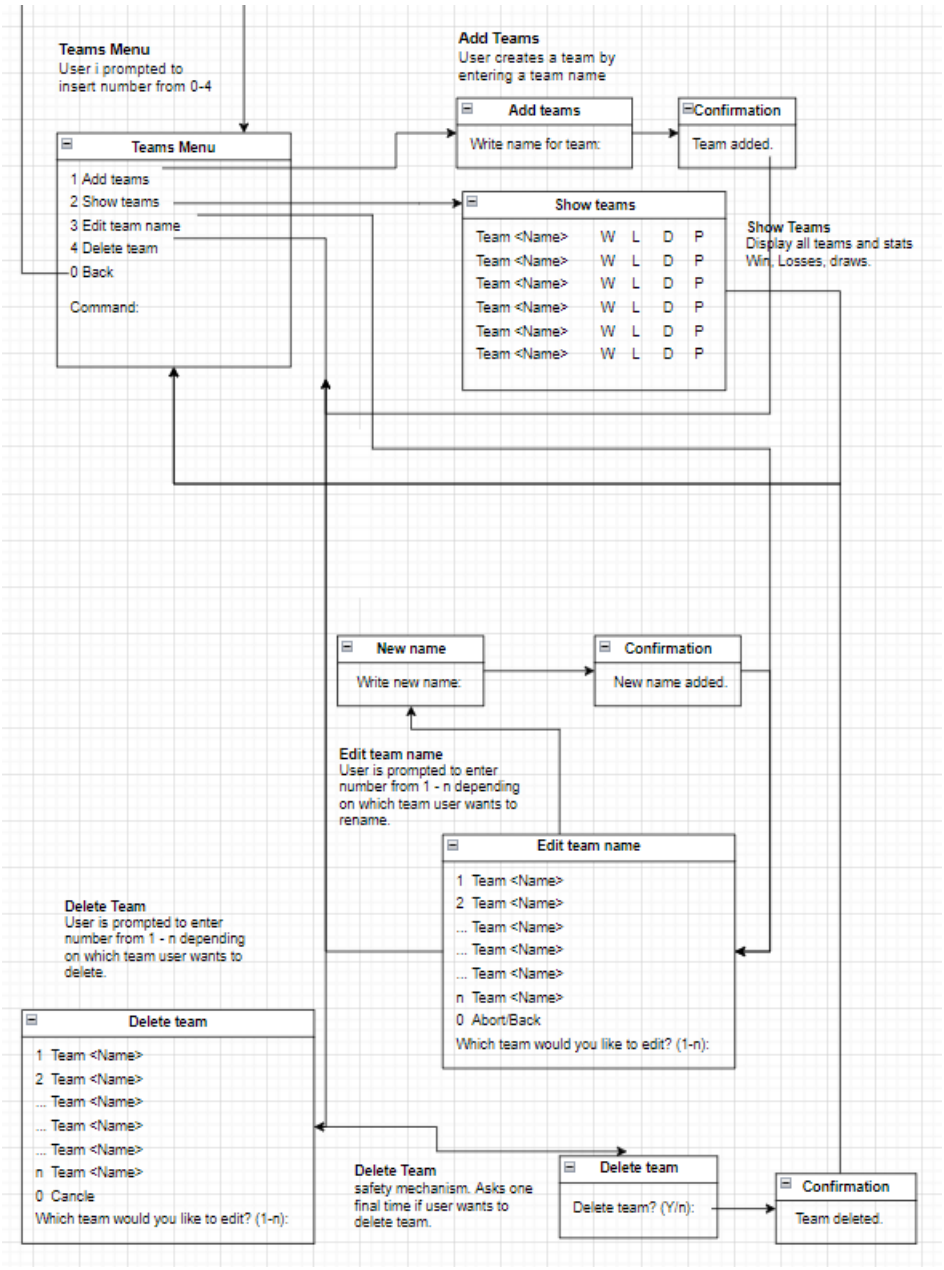


\ *Figure 1 -*

Main menu and create tournament.



\ Figure 2 - Show and edit tournaments.



\ *Figure 3 - Add, show and edit teams.*

home.md

Overview

- Subject: PROG1004
- Group number: 3
- Group members: Andreas Blakli, Arvid Moemeni, Daniel Hinderaker, Jan Olav Lyche Aspelund and Sebastian Instanes Skylstad

Documents

Vision Document \ **Requirements** \ Use-Case diagrams \ Use-Cases (Text) \ Domain model \ Sequence diagrams \ Wireframe \ Universal design \ Usability tests \ **System** \ Project structure \ Class diagram \ Persistence \ Source code \ Installation manual \ User manual \ Testing

sequence-diagrams.md

User, Users Class Sequence Diagram

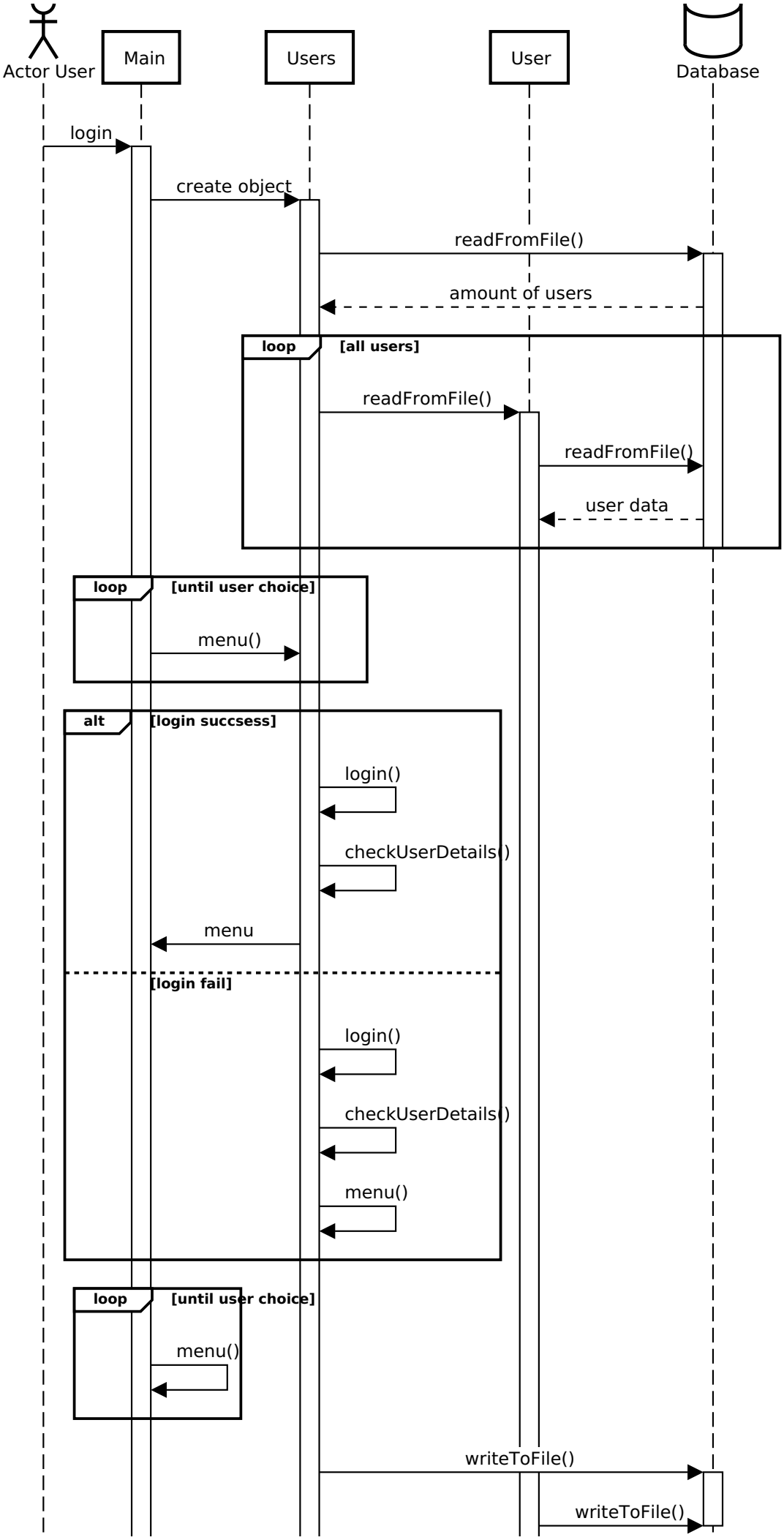
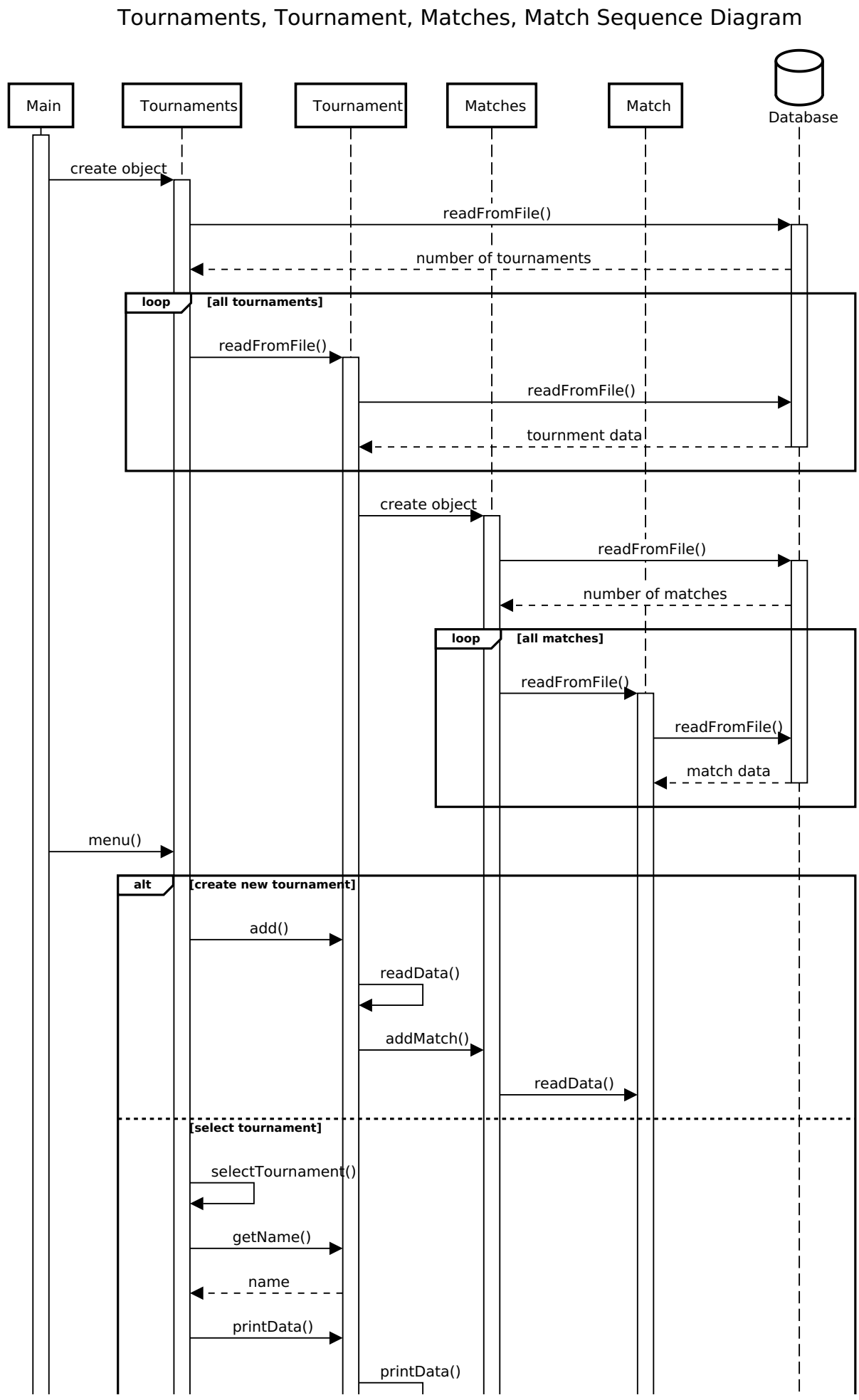

Actor User
Main
Users
User
Database
login
create object
readFromFile()
amount of users
loop
[all users]
readFromFile()
readFromFile()
user data
loop
[until user choice]
menu()
alt
[login succsess]
login()
checkUserDetails()
menu
[login fail]
login()
checkUserDetails()
menu()
loop
[until user choice]
menu()
writeToFile()
writeToFile()

Figure 1 - Sequence diagram of the User and Users classes.



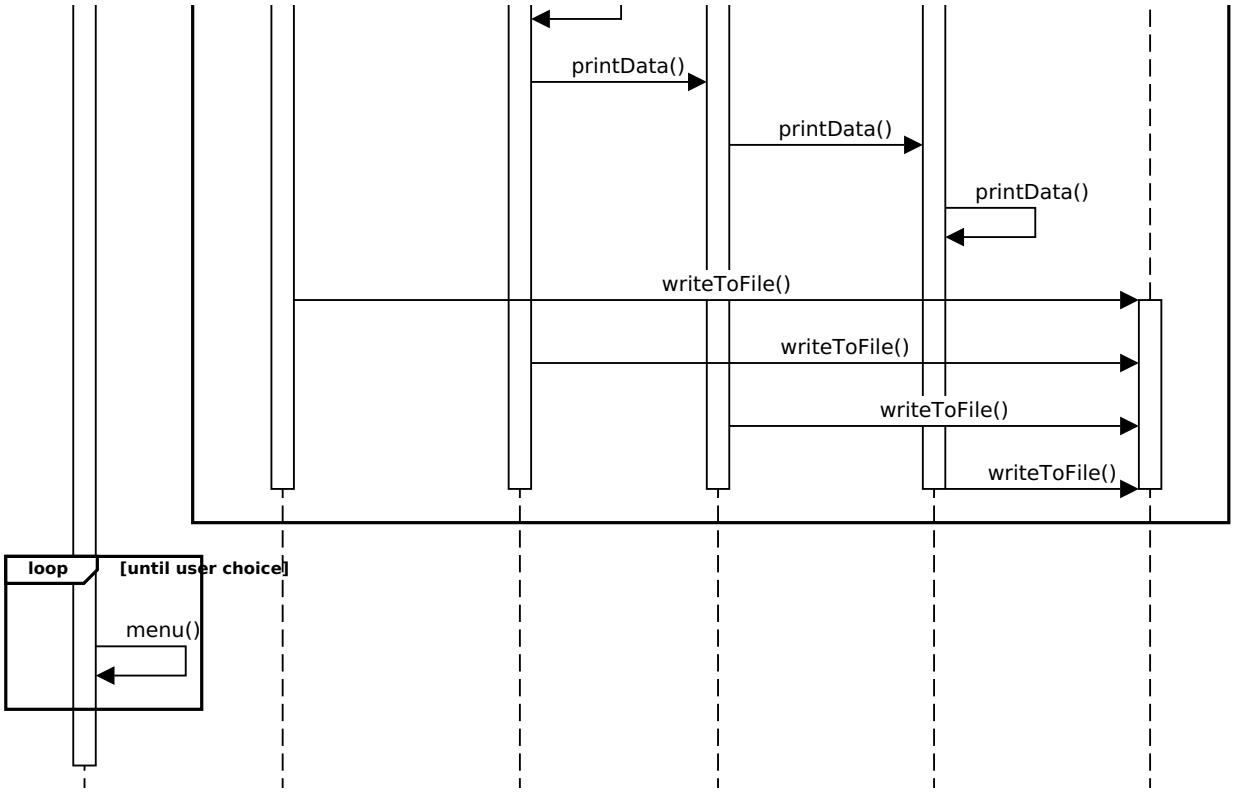


Figure 2 - Sequence diagram of the tournaments, tournament, matche and match classes.

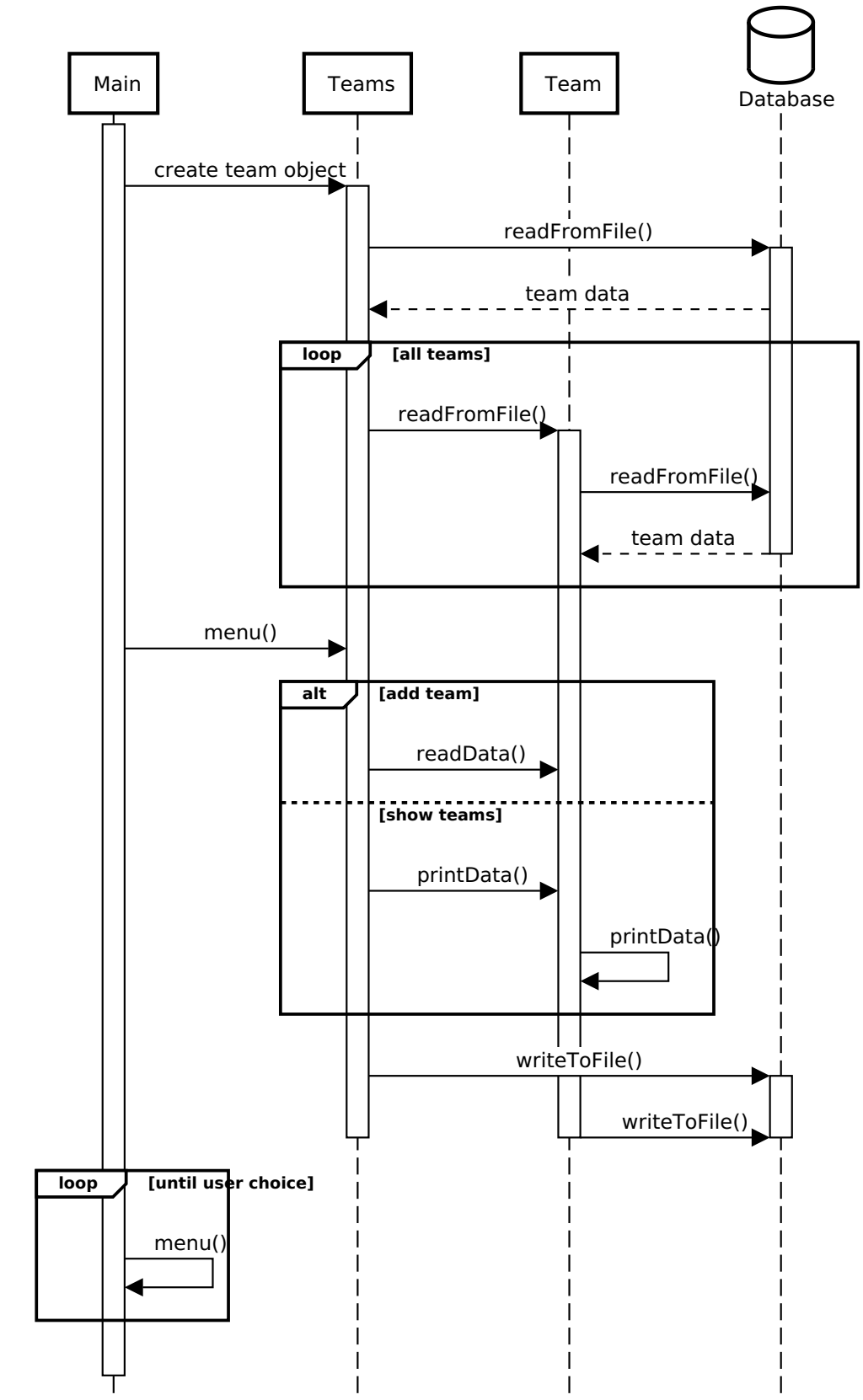
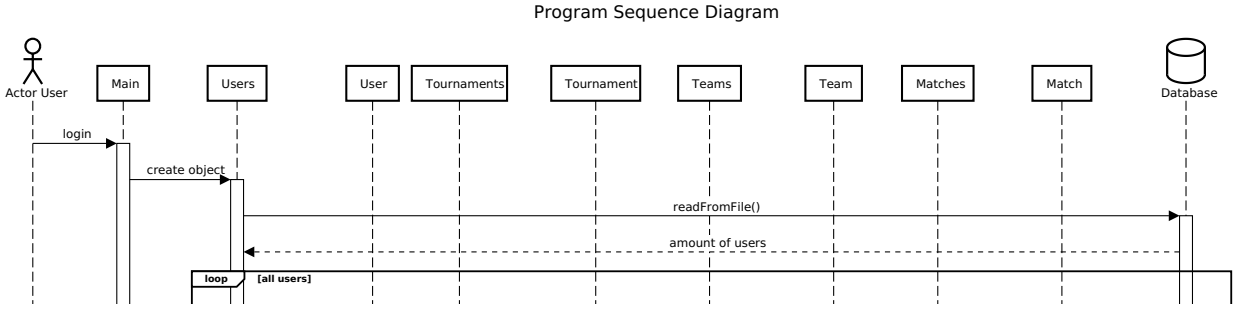
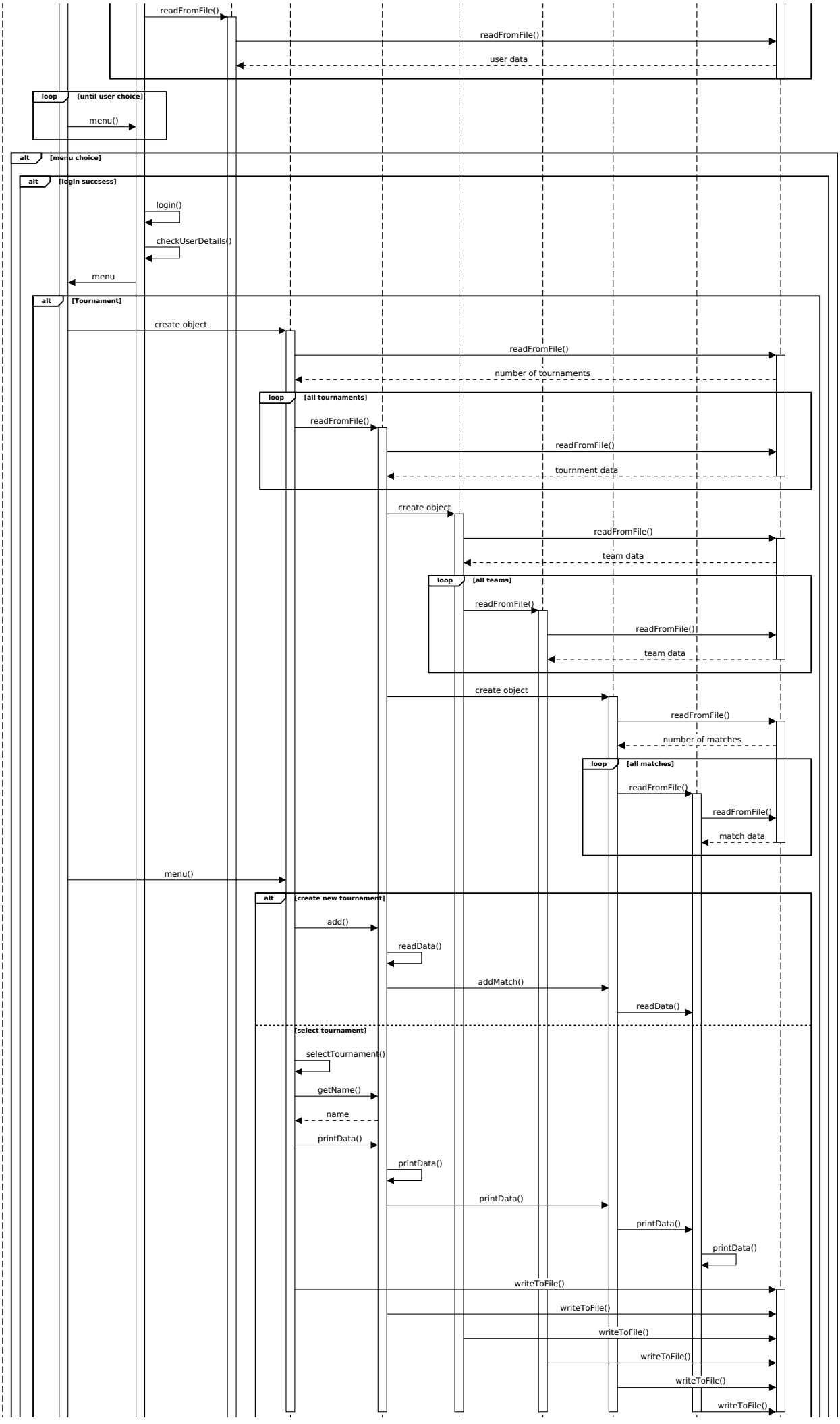


Figure 3 - Sequence diagram of the teams and team classes.




readFromFile()
readFromFile()
user data
loop
[until user choice]
menu()
alt
[menu choice]
alt
[login succsess]
login()
checkUserDetails()
menu
alt
[Tournament]
create object
readFromFile()
number of tournaments
loop
[all tournaments]
readFromFile()
readFromFile()
tournment data
create object
readFromFile()
team data
loop
[all teams]
readFromFile()
readFromFile()
team data
create object
readFromFile()
number of matches
loop
[all matches]
readFromFile()
readFromFile()
match data
menu()
alt
[create new tournament]
add()
readData()
addMatch()
readData()
[select tournament]
selectTournament()
getName()
name
printData()
printData()
printData()
printData()
printData()
writeToFile()
writeToFile()
writeToFile()
writeToFile()
writeToFile()
writeToFile()

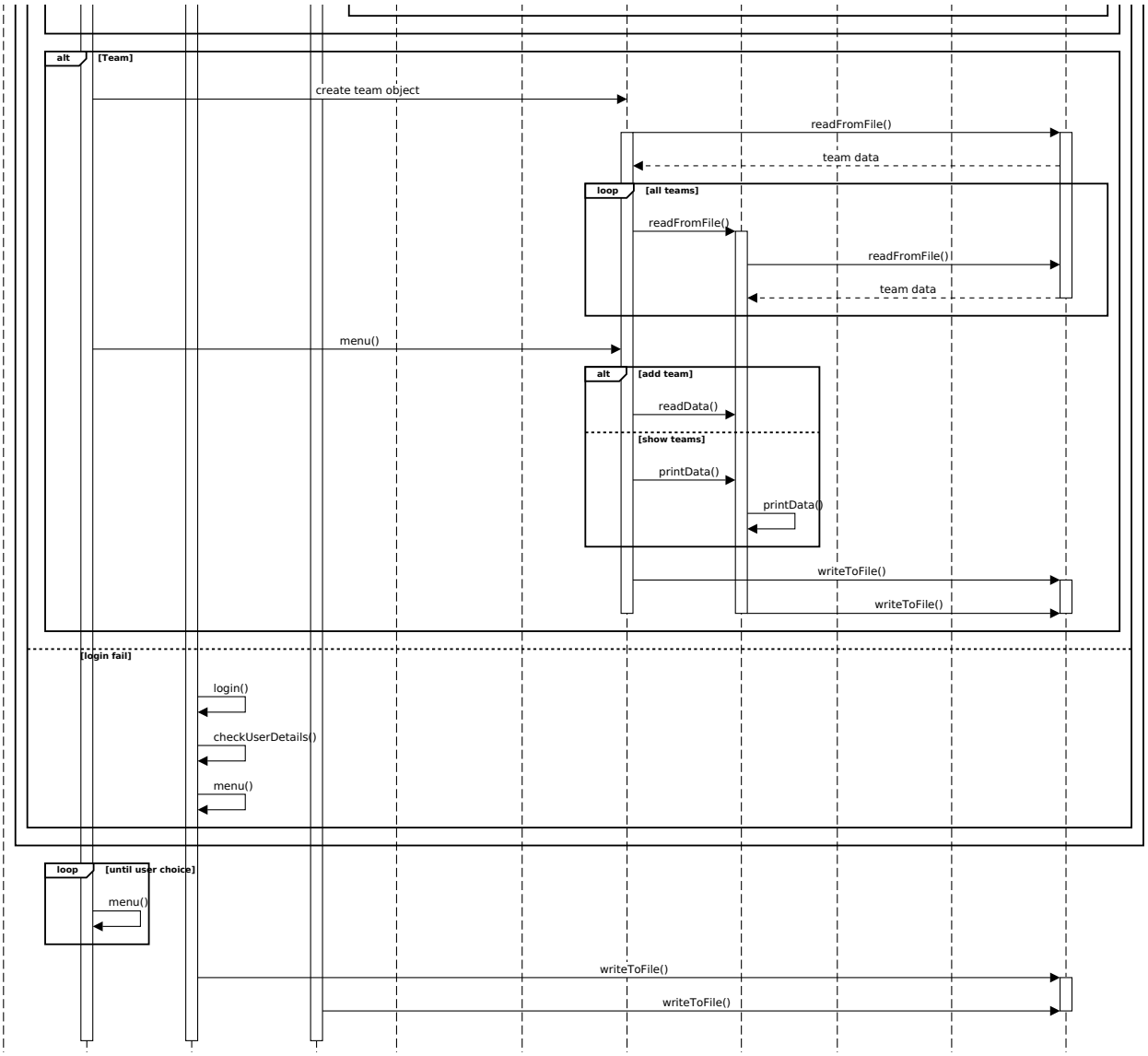


Figure 4 - Sequence diagram of all the classes in the program.