Tournament Generator

Alexander Büchel

200252

Fabio Hilti

200155

Lucy Gannon

200332

Documentation

University of Liechtenstein

Programme: Masters Programme in Information Systems

Module: Information System Development

Course: Information System Development

Assessor: Dr. Frank Breitinger

Working period: 18/09/2020 to 17/12/2020

Date of submission: 16/12/2020

GitHub Link: https://github.com/BuecAle/TournamentGenerator_ISD

Table of contents

Abstract

The following documents describe in detail the process behind the creation of a Tournament Generator for the course Information Systems Development, at the University of Liechtenstein. For the project, students were required to develop a web-based project using Python and Django. There were three students on the team. The process began by brainstorming ideas, and ultimately settling on the idea of a Tournament Generator. The students completed tutorials online to gain an understanding on the technology. When developing the project, the students followed a spiral model of development, which is explained further in the text. These documents detail the approach taken, along with the success and challenges in the process. The project concluded with the successful development of the generator which fulfilled almost all the requirements of the different stakeholders.

Keywords: Tournament Generator; Brackets; Auto create; Requirements;

1 Introduction

On the 9th of October 2020 the first seminar of the course Information System Development (ISD) took place, marking the starting point of the web-based group project. The aim of the project is to develop a web-based project using Phyton/Django. Therefore, skills such as programming, design, creativity, and documentation are required. Before starting with the project, each group member gained a basic understanding of Phyton/Django with the help of various tutorials, online-videos, and the course's assessor itself. This document will be part of the documentation, which includes following methodology: project group, project description, project goals, requirements, domain model, user experience and run through, highlights, readme and notes, and finally, the conclusion.

1.1 Project Group

Our project group consists of three people: Alexander Büchel, Fabio Hilti and Lucy Gannon. Each of us have little to no experience in Phyton, Django, GitHub, or other web-based applications. For this reason, we had to develop our phyton and coding knowledge almost from scratch. Despite our limited experiences, we had big ambitions: we wanted to create a functional, useful web-based project. At first, there were various ideas including simple games such as tic tac toe, or something more complicated like a translation project. Ultimately, we decided to create a Tournament Generator. We made this decision as this was a project that was interesting to us as developers. Further we expected that it would challenge us but that if we worked hard it would not be impossible.

Although each group member tried to work on every project task such as coding, documentation, design, or functionality, we also tried to split up the workload reasonably. Furthermore, our internal communication was certainly more difficult than the years before, due to the global corona-pandemic. For this reason, it was not possible to meet up in person for the project task. The communication occurred mainly online, via Zoom-Meeting or WhatsApp. That is why, we had problems with our communication, planning and organisation.

In the following table below, there is shown, which group member had which experiences beforehand and main responsibilities for the web-based group project.

Table 1: Project Group

Group member	**Experiences**	**Responsibilities**
Alexander Büchel	• No experience in Phyton/Django • Experience in structured text	• Coding • Design • Coordination
Fabio Hilti	• Primarily business background • No experience in Phyton/Django	• Documentation • Coding • Design • Presentation
Lucy Gannon	• Little experience in Phyton/Django • No experience in coding	• Documentation • Coding • Presentation

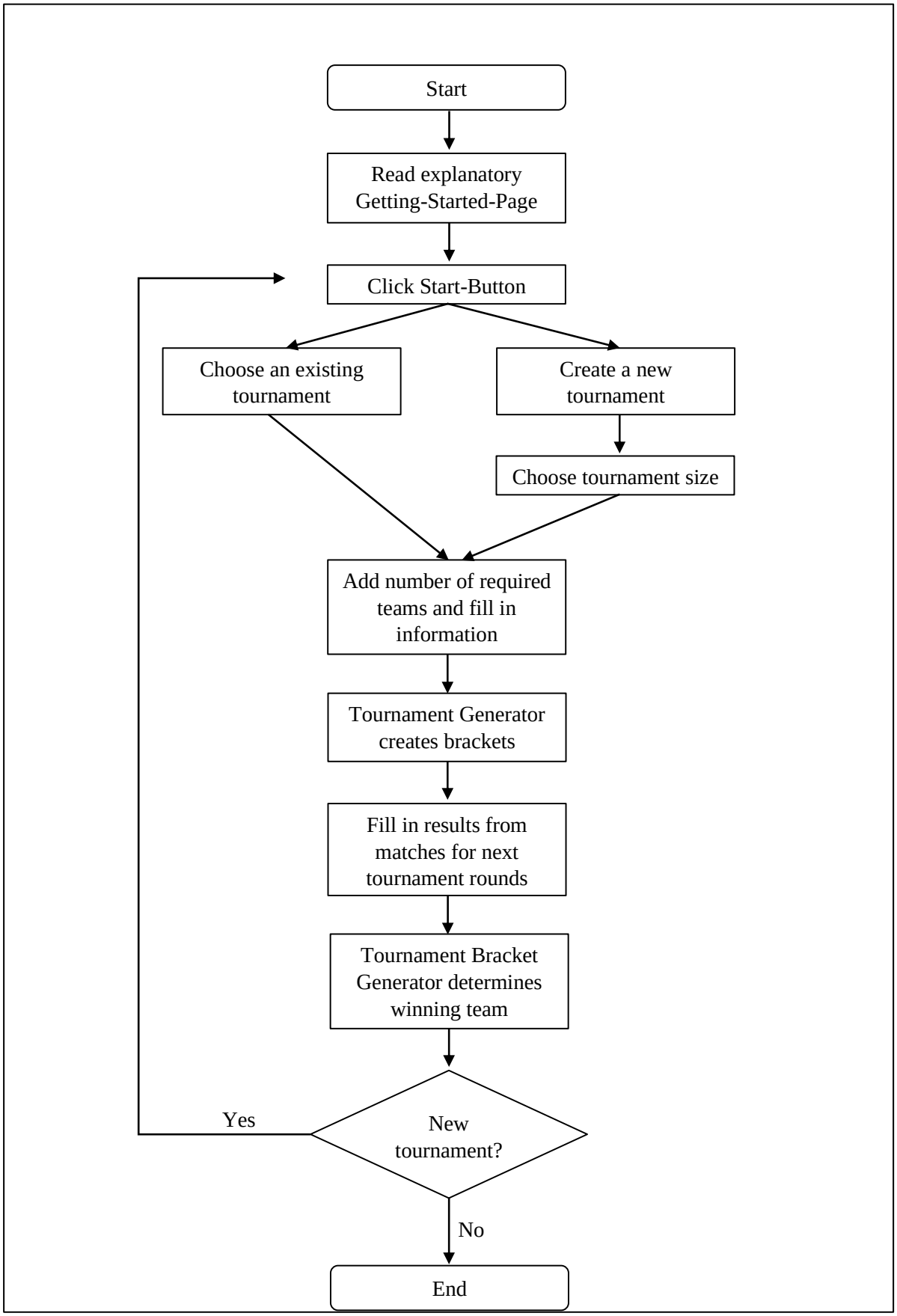
1.2 Project Description

Everything can be a competition. To compete with one and each other is in the human nature – it is fun and challenging. All of us three in our group like to participate in different competitions, respectively tournaments in a sport club or just for fun with friends. With that thought in our mind, we developed the idea to create a "Tournament Generator" as our web-based group project in the module "Information System Development" with Python/Django.

With this project, a group of friends, respectively a sport club can simply create a tournament tree/tournament bracket without any complicated prior knowledge. There are already some bracket generators to use on the world wide web, but most of them have unnecessary customization, which will take up a lot of time for the end-user and can be quite tedious. That is why, we as a project group, decided to create our own simple "Tournament Generator" without beating around the bush. The key elements of this web-based project are the three different tournament brackets, which the end-user can choose from: 8, 16 or 32 teams. This means, depending on the tournament size, the tournament bracket will alter themselves on the required size.

For a simple demonstration and understanding on the basic procedure of our web-based project "Tournament Generator", we created the following flow chart on the following page (page 3). This provides a substantial description of our project.

Figure 1: Flow chart Tournament Generator



2 Project Goals and Target User

The goal of this web-based group project is to create a basic Tournament Generator, which – as its name already says – generates different brackets for tournaments with a size of 8, 16 or 32 Teams. In Addition to the brackets/tournament trees, the end-user of should have the possibility to create different teams for their tournaments with several information fields such as team name, number of players, manager, and captain. This teams can be added anytime to the desired tournament.

As previously stated, the aim is to create a Tournament Generator for people, who like to be competitive. Hence, the main target group we would like to address with our web-based project are mainly people, who want to organize a spontaneous sporting event without much planning for sports such as football, tennis or handball etc. Moreover, our web-based project should appeal not only for sport-users but also for competitive fun-tournaments in general (e.g. gaming, Beer Pong, other competitive party-games).

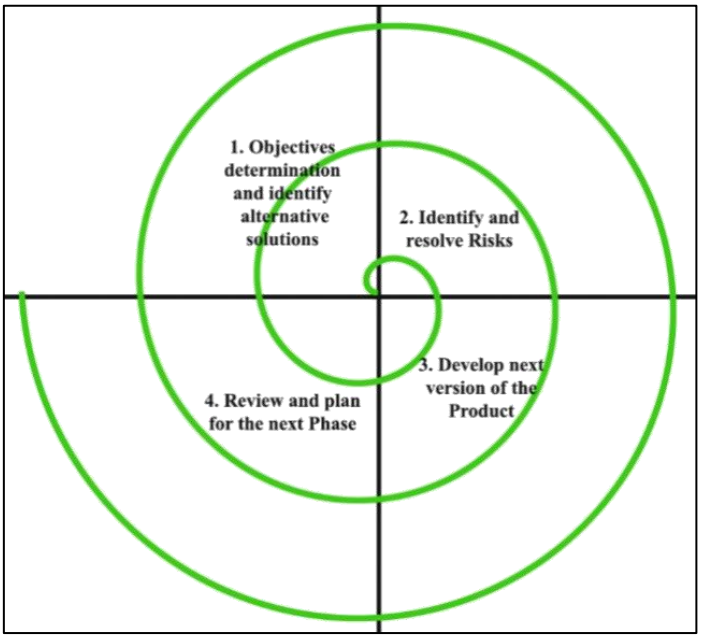
With our own "Tournament Generator" our goal is to offer a basic but interactive website, which will create brackets for the three common tournament sizes that can be used and implemented easily in real life.

3 Requirements

With the project goal in mind, the need to define specific requirements arose. For defining the requirements and process of the "Tournament Generator", the spiral model for Software Development is used (see Figure 2). The goal of creating requirements is mainly to minimise the risks and gain satisfaction of stakeholders.

The spiral model provides support of risk handling, whereby each loop is called a phase of the software development process. The exact number of phases depend on the group and the project work itself. The 4 phases are "objectives determination and identify alternative solutions", "identify and resolve risks", "develop next version of the product" and "review and plan for the next phase" (Kumar Pal, 2018).

Figure 2: spiral model



3.1 Stakeholder requirements

As a first step, it is needed to describe the potential and different stakeholders, which also usually have different requirements for the project. For our own web-based projects, the following 3 stakeholders were identified:

1. Dr. Frank Breitinger, Assessor of course Information System Development (ISD)
2. End-users of Tournament Generator
3. Project Group

3.1.1 Requirements Dr. Frank Breitinger

- A certain minimum of functionality and complexity of the project/website
- Clear documentation of the project including docstrings, readme and examples of usage
- Nice Layout and Design for the project/website
- Code quality of project is well structured and organized
- Chosen project with its features and ideas behind is creative
- Utilization of GitHub or any other control system
- A list of used packages including version and used programming language
- Every member in project group contributes to coding

3.1.2 Requirements end-users

- Web-based project is optimized for all common Internet-Browser such as Microsoft Edge, Google Chrome, Safari and Mozilla Firefox
- Web-based project is simplistic and easy to use and understand, respectively user-friendly
- Web-based project is able to create the three common, different tournament sizes (8, 16 or 32 teams)
- Web-based project allows to create teams, which can be added to various tournaments, including the information fields: team name, number of players, manager, captain)
- Web-based project includes information about the creators of the website and an imprint
- Web-based project guides the end-user on how to easily create a tournament
- Process of the web-based project is simple, quick, and efficient
- Web-based project has a uniform design and is appealing for the end-users
- Web-based project offers a random-autofill, which fills up needed fields with information automatically for an even faster tournament creation
- End-user can put in results of the different matches for deciding the winders of each tournament round

3.1.3 Requirements project group

- Project group achieves greater understanding in coding, especially in Phyton/Django
- Project group has good internal communication and achieves a great working environment
- Each member of the project group takes responsibility

- Each member of the project group participates in required project tasks such as coding, documentation, or design
- Project group splits the workload of the web-based project fair and evenly
- Different tasks have a common theme (e.g. uniform design, structured GitHub)

3.2 Functional and non-functional requirements

In addition to the previous stakeholder requirements (see 3.1.), the requirements for our web-based "Tournament Generator" can be categorized in functional and non-functional requirements.

Functional requirements are statements of services the project should provide, how the system should react to specific inputs and how it should behave in particular situations. Furthermore, it can also state, what the system specifically should not do. Non-functional requirements are constraints of the functions such as standards, timing, programming language or development process. They often apply to the system as a whole rather than individual features (Sommerville, 2011). The following functional and non-functional requirements are identified in chapters 3.21 and 3.2.2.

3.2.1 Functional requirements

- Tournament Generator lets end-user create its own individual tournament
    - Create new tournament or choose existing tournament
    - Minimum and Limitation of teams per tournament (until reached full capacity)
    - All created tournaments are displayed on web-based project/website
- Tournament Generator should include a logical path/process from the Creation to the Execution of the created tournament
    - Step-by-step explanation at Starting-Page
    - Process runs like a common thread through tournament creation
- Tournament Generator must include the creation of 3 various tournament brackets
    - Tournament bracket for 8 teams
    - Tournament bracket for 16 teams
    - Tournament bracket for 32 teams
- Tournament Generator should display the winner at the end of each tournament
- Tournament Generator allows end-user to put in match-results to choose the winner of each round
- Tournament Generator enables the creation of teams for tournaments
    - Add or delete teams in tournaments
    - Add information to created teams such as team name, number of players, manager and captain
    - All created teams are displayed on web-based project/website
- Tournament Generator allows random Autofill-function for an even faster creation of tournaments and teams
    - Autofill information for team creation
    - Autofill information for tournament creation
- Tournament Generator contains an About-page
    - Short description of project group

    - Contact data of three group members such as e-mail, address or telephone
- Tournament Generator contains an Imprint-page
- Tournament Generator has a uniform Corporate design
    - Contains Footer
    - Contains Navbar
    - Contains Logo
    - Contains Buttons
- Tournament Generator must be precisely documented
    - Documentation
    - Docstring and comments in Phyton/Django project
    - Readme on GitHub
    - Examples of Usage
- Tournament Generator is shared on GitHub
    - Shared GitHub branch contribution
- Tournament Generator adapts to different common Internet Browsers
    - Microsoft Edge
    - Google Chrome
    - Safari
    - Mozilla Firefox

3.2.2 Non-functional requirements

- Tournament Generator has appealing and trendy design in the eyes of target end-user
    - Modern and uniform design
    - Neatly arranged and organized
    - Structured web-based project/website
- Tournament Generator must have a high-quality standard and good usability
    - Few to no errors/bugs
    - Adaptability
- Tournament Generator must be easy to maintain and easily usable for end-user
- Tournament Generator has a good data security
    - Inputs from end-users are safe
- Tournament Generator code is well-structured and logical
- Tournament Generator documentation is clear, complete and accurate
    - Documentation on paper
    - Documentation on Phyton/Django via commentary/docstrings
    - Shared GitHub
- Tournament Generator has a good performance and scalability
    - Web-based project can handle higher data input
    - Web-based project returns results fast
- Tournament Generator must have a coherent overall image/impression

3.3 Requirements validation

According to Sommerville there are four different keywords for the checking of the requirements: validity, consistency, completeness, realism, and verifiability (2011). With these validation-keywords our "Tournament Generator" will be reviewed with the question in mind if we have fulfilled the mentioned requirements.

Table 1: Requirements validation

Keyword	**Clarification** *(Sommerville, 2011)*	**Validation**
Validity	Does our system provide the functions which support our user's needs?	Yes, we almost provided every required function. Although, we could not develop the tournament trees/brackets as we initially intended
Consistency	Are there any requirements, that conflict themselves?	No, there are not any requirements that highly conflict themselves. One thing is to mention, that not all Internet Browser show the Tournament Generator the same way, we initially intended.
Completeness	Are all requirements for the project included?	All major requirements for the Tournament Generator are more or less included.
Realism	Can the requirements be implemented with the available budget and technology?	Yes, the requirements can be implemented. For a simple tournament bracket, there is little to no budget as well as special technology besides the standard needed.
Verifiability	Is it possible to check the listed requirements?	Yes, the requirements can be checked, when creating a tournament with our Tournament Generator website.

The following functional or non-functional requirements were partially met or not met at all:

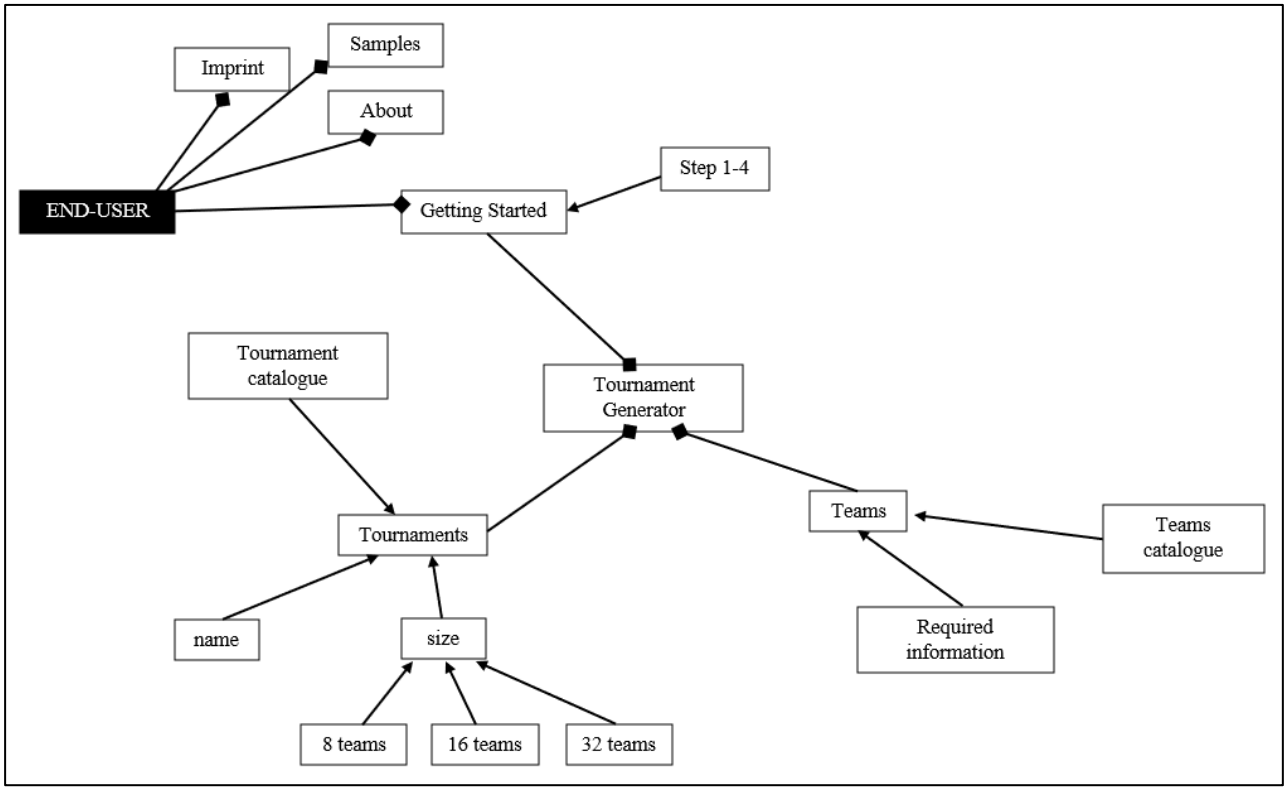
- Design and Functionality of the tournament brackets for 8, 16 and 32 teams are not on the intended level we initially considered. Instead of automatically choosing the winning team for the next bracket, the end-user has to choose the teams via dropdown-button.
- There are Problems with the Footer positioning and depiction for the Tournament and Team creation pages, which blocked the view of the content blocks.
- Shared GitHub gives out error-messages due to db.sqlite3, which sometimes leads to difficulties with updating/push/pulling.
- Different, individual representations of the Tournament Generator website at the individual internet browsers. Some styles are not always displayed correctly or not at all.

- Design of the website does not always remain consistent.
- There may still be some bugs present with intensive use. However, the process mostly runs smoothly without any bugs or error messages.
- Result input is not integrated – Workaround with choosing the winner of the last stage

4 Domain Model

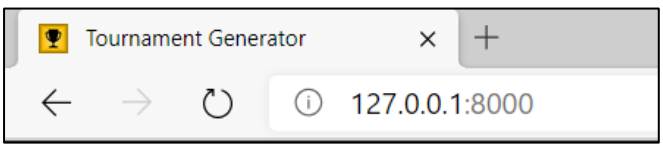
For an optimal and simple overview of the end-user's most important functional requirements of the web-based project, we created a domain model, which is depicted below (Figure 3). For the creation of the domain model we orientated oneself by the example of Rosenberg and Stephen (2007).

Figure 3: Domain model for Tournament Generator

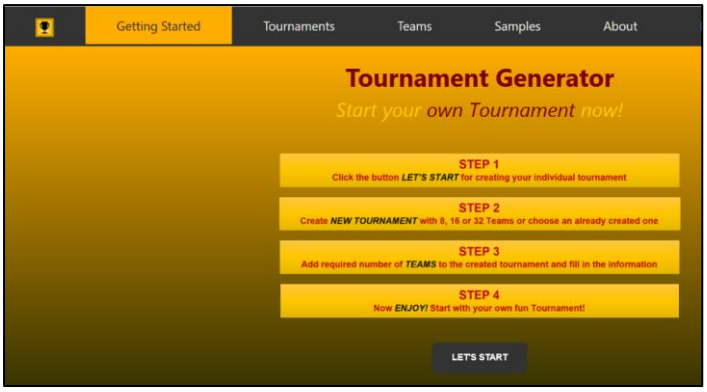


5 User Experience and Run-through

With the help of screenshots and text, we would like to show you the step-by-step process of our web-based project. The best user experience can be delivered by using Google Chrome as browser.

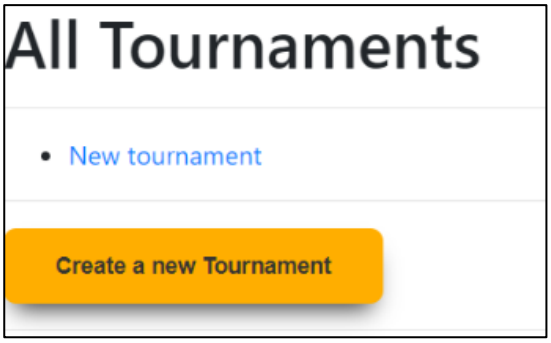


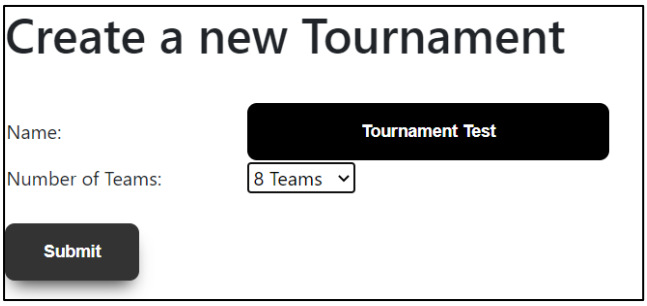
1) With the standard server/ internet address http://127.0.0.1:0000 you can get access to the Tournament Generator website (as long as the server is running). The official logo of our web-based project "Tournament Generator" is a black trophy in a square with an orange fading background. It can be seen on the internet tab.



2) The landing page / home page is also the getting started page. It contains the title, slogan and a short step-by-step guide on how the end-user can create his own individual tournament. With a click on the button "Let's Start" the end-user starts the process of tournament creation.

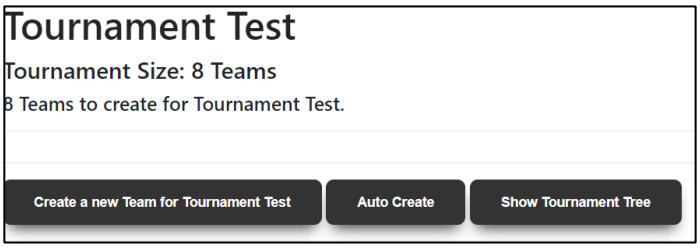
3) After clicking the button on the Getting Started page, you can access the overview of all created tournaments. Now you have the possibility to select an already created tournament (in this example this would be "New Tournament") or to create a completely new tournament by clicking on the lower button "Create a new Tournament".

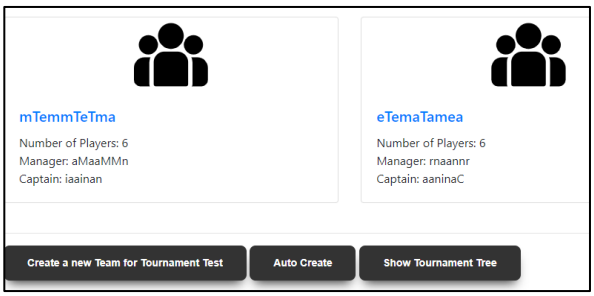




4) So, the next step in this run trough is to create a completely new tournament. For this example, we will name it "Tournament Test" and select a tournament size of 8 teams. There are also 16 or 32 teams available. After we have completed these steps, click once again on the "Submit" button at the bottom.

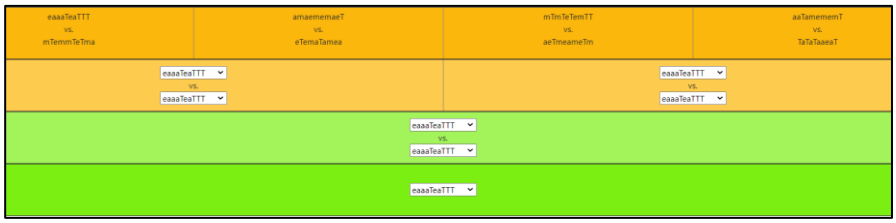
5) Now you get to a new interface where you have to create the associated teams. Either you create a new team individually and enter the name, number of players, captain and manager yourself or you select the "Autocreate" button, which fills in these fields automatically. At the top, below the title, you can see how many teams are needed for the tournament (in this case the end-user needs to create 8 teams). When all teams are created the Tournament Generator says, that all required teams are created.

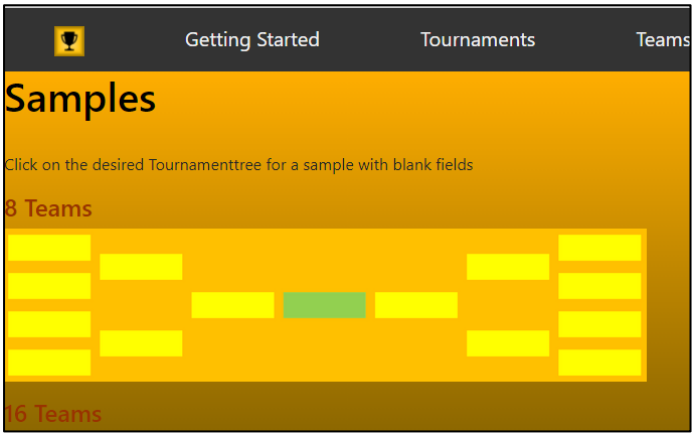




6) Now you get an overview of all created teams, which should participate in the individual tournament. In this example they were created with the function "Autocreate". If the end-user is satisfied with the settings, he/she can continue by clicking on the "Show Tournament Tree" button.

7) The generated tournament tree is now displayed (here in the example for 8 teams). The first matches are listed in the top orange field. Using dropdown buttons, the respective winner of the games played can be determined until the overall winner remains at the end.





8) After the winner is determined, the tournament is finished and thus the Tournament Generator has fulfilled its task. The end user can either create a new tournament or download a tournament tree in the sizes 8, 16 or 32 teams via the page "Samples". These are empty sample brackets. Other pages on the Tournament Generator website are "About" and "Imprint", but these are not related to the creation of a tournament tree.

6 Readme and Notes

An explicit documentation/commentary of the used code in our web-based project “Tournament Generator” is used with docstrings in Python/Django itself. In Addition to the commentary, there is a Readme available on our shared GitHub.

GitHub Link: https://github.com/BuecAle/TournamentGenerator_ISD

7 Conclusion

To conclude the documentation, we would like to record in what we think we have succeeded in doing and what we have done rather less well

7.1 What went well?

- **Fulfilled requirements for all major stakeholders:**

As detailed above, we strongly feel we met all the requests of the stakeholders in this project.

- **Aesthetic and Intuitive Design:**

The Tournament Generator is minimalist which is pleasing to the eye. Further – and perhaps more importantly design wise – it is easy to use. A long explanation is not needed as the design aids the user in understanding the process.

- **Code Quality**

The code is clearly structured. Created apps tournament and team make it easy to understand. Every file of the project contains comments which describe the meaning of the code. Each row has been indented, making the code easier to read.

7.2 What went wrong?

- **Communication**

Given the situation with Covid-19 all our communication was online. This made completing the project slightly more difficult as we were unable to meet to discuss the project. Misunderstandings are less likely in person.

- **Differences in Skill**

Not all the team members were as able and adaptable as others. This slowed down the progress of the project.

- **Specific Coding Challenges**

Finding the correct code for generating the tournament tree took much more time than expected. Several different methods were trialed. Ultimately, we did find a method, as described above.

In retrospect, having finalized the project and we have evaluated what went well and what went rather wrong. As mentioned in the introduction part of the documentation, this was our first project where we worked with a program language such as Python/Django. All things considered we are pleased with what we achieved. We are confident we have overcome the major difficulties with the project. There are always things to improve and we hope we will get even better for the next web-based project.

List of references

Kumar Pal, S. (2018). *Software Engineering/Spiral Model* [Diagram]. Geeksforgeeks. https://www.geeksforgeeks.org/software-engineering-spiral-model/

Sommerville, I. (2011). Software Engineering (9th ed.). Boston, MA: Pearson

Rosenberg, D., & Stephens, M. (2007): Use Case Driven Object Modeling with UML: Apress.

Tournament Generator

Declaration of authorship

We hereby declare that the present paper is entirely my/our own work and without the use of any unauthorised assistance. Any content which has been taken verbatim or paraphrased from other sources has been identified as such. This paper has not been submitted in any form whatsoever to an examining body. Previously published work has been cited as such.

Vaduz, 16.12.2020

Alexander Büchel

Fabio Hilti

Lucy Gannon