FMT: Conference Support System

Design Project Report

University of Twente

Group 6:
Aleksandar Petrov
Benjamin Othmer
Daniel Mocanu
Jiayi Tan
Kristiyan Michaylov

Supervisor: Prof. Dr. Marieke Huisman

Date: 21st April 2023

1 Abstract

Conference support systems are applications that facilitate the organization of joint conferences. The European joint conferences on theory and practice of software (ETAPS) are a yearly event comprised of multiple conferences occurring simultaneously over a period of several days. The chairs of the individual conferences as well as the overall steering chair of the joint conferences are responsible for organizing and planning the conference schedule and selection of the individual conference committees. The overall steering committee chair of ETAPS has the responsibility to outline the schedule and finalize (and publish) the schedule and committees after verifying that no presenters are scheduled to present in two conferences at the same time and day and that committee members are exclusive to a single conference. Our client provided us with the project to create an application that will facilitate this process, since the procedure and information exchange was conducted using Google Sheets, which does not meet the requirements for an ideal exchange of information and therefore costs manual effort for both the overall steering committee chair and the chairs of the conferences.

Our solution is a RESTful web-application, tailored to the specifications and requirements of the stakeholders. Additionally, to outlining, scheduling, and proposing committee members, the web-application features the functionality to import CSV-files from the EasyChair website, export data in a proprietary YAML format (for displaying the schedule on the ETAPS website) and mechanisms to assist the steering committee chair in finalizing the schedules and committees by indicating possible issues (such as members being in multiple timeslots and committees).

2 Table of Contents

3 Glossary

In order of occurrence

Abbreviation	**Definition**
ETAPS	European Joint Conferences on Theory and Practice of Software
ESOP	European Symposium on Programming
FASE	Conference on Fundamental Approaches to Software Engineering
FoSSaCS	Conference on Foundations of Software Science and Computation Structures
TACAS	Conference on Tools and Algorithms for the Construction and Analysis of Systems
SPIN	International Symposium on Model Checking of Software
PC	Program Committee
SC	Steering Committee
OSC	Overall Steering Committee
SCC	Program Committee Chair
OSCC	Overall Steering Committee Chair of the ETAPS conferences
CC	Conference Chairs of the individual conferences
API	Application Programming interface

4 Introduction

The European Joint Conferences on Theory and Practice of Software (ETAPS) is a confederation of multiple conferences and symposia: the European Symposium on Programming (ESOP), the International Conference on Fundamental Approaches to Software Engineering (FASE), the International Conference on Foundations of Software Science and Computation Structures (FoSSaCS), the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), and the International Symposium on Model Checking of Software (SPIN) – which has been collocated with ETAPS in the year 2023.

The ETAPS conferences occur yearly, with over 500 researchers participating. Each individual conference has its own Program Committee (PC) and Steering Committee (SC) and is governed by the overall Steering Committee (OSC) of ETAPS. Typically, the ETAPS conferences have individual programs for each of the 4 days that the event takes place, but some exceptions occur, i.e., for plenary sessions, introductions and workshops are shared events for multiple conferences.

Every conference committee has a Chair – the SC's have steering chairs (SCC), with multiple responsibilities, including the creation of the programs and program committees and of the respective conferences in conjunction with the Overall Steering Committee Chair of the ETAPS conferences (OSCC). Multiple factors must be considered during this process: The program committees must be created without overlap and the program schedules cannot involve the same speaker(s) in two conferences simultaneously (unless in plenary sessions et cetera). Additionally, the invitation and paper acceptance process are being conducted with EasyChair – a web-based conference management software system. Finally, the program schedules and committees are published on the ETAPS website itself, after the OSCC finalizes the data, exports it, and formats it for publication on the website [1].

4.1 Current Solution

The current solution utilized by the client, the OSCC, involves Google sheets for storing, editing the data, and creating schedules as well as verifying that the individual conference committees have no overlaps. Besides the tedious manual checks involved in the process of analysing data, other problems that arise in the current solution include inconsistency in the format of the stored data, difficulties in visualizing and analysing the data and problems in keeping track of the changes that occur.

4.2 New Solution

The client has different tasks connected to their job as part of the OSCC of the ETAPS conference. To satisfy the needs of the client, the development team implemented 2 systems that ease and solve most of the inconveniences the client experiences in the current situation., with one of the important aspects being to design a user-friendly interface with a minimum number of clicks (the intention is to minimize the amount of tedious and repetitive work our client needs to do).

4.2.1 System 1: ETAPS conferences program schedule

The first system includes a front-end and back-end that communicate the data through API, designed by the development team. The main responsibilities and functionalities for the conference schedule include allowing the OSCC to create a multi-conference schedule outline and assign slots to the individual conferences. The individual CCs can then fill the program outline and assign slots with speakers and topics. Then, the system compiles an overview of the programs indicating possible overlapping schedules for each conference or duplicate entries for each conference, allowing the OSCC to make changes and finally submit the programs. Finally, the program can be serialized in YAML format, which is usable by the ETAPS website to display the conference programs.

4.2.2 System 2: ETAPS conferences committee selection

The second system is similarly designed and structured to the first one. However, the main function of system 2 is to allow the individual CCs to enter the data of proposed committee members and the OSCC to validate it, by editing and inspecting the data provided by the CCs, so that consistency and correctness can be ensured. Then, system 2 generates the information in a specified format: “FirstName” “LastName” <E-Mail> which is required as input to the Easychair website to manage committee members' invitations. Finally, System 2 facilitates the process of members accepting or rejecting the invitation on Easychair and to allow the users to make changes to the committees before finalizing it.

4.3 Project Scope

4.3.1 Scope

The final project includes the final products of the two systems. As it was agreed in the first two meetings, the development team was responsible for designing and implementing the front-end, back-end and database parts, while utilizing solutions which are robust and reliable. Therefore, the client was not interested in the technical specifications of the chosen tools/frameworks, rather in the ability to solve her current problems. Furthermore, given the nature of the project and the idea to be used by people, different from to the client, the development team provided a detailed report, explaining design choices, API structure, requirements, user stories and all the details related to the technical part of the project. Moreover, a clear and concise manual is provided. It could be used from new and possibly more experienced people to familiarize themselves with the possibilities that the respective systems supply.

4.3.2 Exclusions & Constraints

The two systems offer solutions for the existing problems that our client was currently facing. Nevertheless, as every software product, there have been several constraints.

First and foremost, it could not be guaranteed that the final systems will be easily extendable for the future needs of ETAPS. Our team was focused on implementing all the “must” requirements that were elicited during the second meeting with the client and none of them specifies that the solutions should be easily extendable.

Second, since both systems were implemented as a web application, it was agreed with the client that working on Firefox is under the category “must”, whereas Chromium based, and Safari are

under the category of “should”. This might cause an inconvenience for some users; however, the types of browsers are different, thus in some cases they render/show the same content in various ways.

Third, the webpage design of the 2 systems is not required to scale to mobile devices. Although not ideal, our client has informed us that the system is mainly accessed from a laptop or a personal computer, therefore this is a rather minor inconvenience.

Finally, the final product is hosted locally from our side. It was discussed with the client that our team is not responsible for hosting the systems publicly, since that is out of the scope for the project. Furthermore, any work outside the end of the design project module, related to maintenance, optimization, bug-fixing is outside the scope of the project and will be avoided by the development team.

5 Project Organization

5.1 The Team

Aleksandar Petrov: 3rd year TCS student from Bulgaria. His interest are sports, algorithms, and mathematics. In his third year he was Teaching Assistant for 2 modules. His role in the project is to be a part of the designing and developing the front-end of the application.

Benjamin Othmer: 3rd year TCS student from Germany. His interests include information technology, cyber security, sports and more. Ben has had the chance to gain project-related experience in his student job in the last year and aims to apply the knowledge gained thus far to the project. His role in this project is to design and develop the back-end as well as empower the team to keep on track with the schedule.

Daniel Mocanu: 3rd year TCS student from Moldova. Is interested in sports, strategy games, algorithms, and mathematics. His role in the Conference Support System project is to be one of the front-end developers with a fallback to back-end in case the back end gets behind in development.

Jiayi Tan: 3rd year TCS student from China. Her interests include cyber security, autonomous system, sports and more. Jiayi finished her 1.5-year Bachelor Honours programme of mathematic track in her 2nd year and has been a Teaching Assistant until now. Her role in this project is to design business logic and integrate front-end and back-end components as well as evaluate the system security.

Kristiyan Michaylov: 3rd year TCS student from Bulgaria. His interests include technology, programming, books, sports and more. Starting from his 2nd year, Kristiyan has been a Teaching Assistant for several modules. This experience helped him improve his communication and analytical skills. For the Conference Support System, he would be one of the people responsible for the front-end, and in addition serving as client spokesperson.

5.2 Roles and Responsibilities

Name	Role/Responsibilities
Jiayi Tan	Overall system design Front- and back-end integration Back-end Security officer
Kristiyan Michaylov	Front-end development Client spokesperson and external communication Project Management
Daniel Mocanu	Front-end developer Security researcher Front-end security implementer Front-end designer
Aleksandar Petrov	Front-end development Front-end design Front-end research
Benjamin Othmer	Back-end development Back-end research Project management Database architect Database researcher

Table 1: Team Roles and Responsibilities

The project responsibilities have been separated into two major components.

The application component consists of four overall categories:

I. System Design and Database: This category was responsible for designing, updating, and integrating the overall application (back and front-end). The former is a high-level view on how the system functions as an orchestra of its individual components and how they will interact on a logical level.

II. Front-end: This category was responsible for the application views and user interaction. These tasks consist of designing the interface including user interaction, application logic (how the data will be presented and manipulated by the user). Additionally, the front-end will have a high interactivity and integration with the back-end, hence a lot of emphasis is put on the latter.

III. Back-end: The primary purpose of this category was to provide an application programming interface between the front-end and the database. The back-end also is the control component of the overall system and provided security and data processing functionalities. Due to the flexible nature of our back-end a lot of emphasis was laid on the front- and back- end integration, so that the API can be expanded upon (i.e., including data processing and parsing components).

IV. Database: The database category is responsible for designing and facilitating the data in a structure that empowered the back- and front-end to perform their tasks.

The project organization and communications component consisted of two categories:

I. Project Management: This category was responsible for ensuring that the team adheres to the schedule or update it if necessary. Additionally, this role is intended to ensure cooperation between the individual application roles categories. Finally, this category was primarily responsible for preparing meetings and empower the team to facilitate the workflow and application and ensure that the deliverables are completed.
II. Spokesperson and external communication: This category was connected to the primary spokesperson between the group and client. It was decided that it is more efficient to let the information flow through this category, because confusion and redundancy could be avoided. Finally, this category was responsible for keeping the client up to date on the progress of the project (outside of meetings).
III. The responsibilities were subject of change and no person was solely responsible for a single task. The main purpose of this distribution was to guarantee accountability for every component and category, so that nothing gets left behind and so that every member in the team has a spokesperson.

5.3 Risk Analysis

There are some potential risks that could arise during the development of the systems. In every project there are certain risks that could prevent a successful outcome. These risks must be identified, analysed, and mitigated to reduce the chance of a project failure. Various studies and meta-analysis have been conducted to identify the most common risks in software engineering projects. In 2003 IEEE Software published an article titled “Why the Vasa Sank: 10 Problems and Some Antidotes for Software Projects” [2], providing an analysis of the biggest problems (and some antidotes) occurring in software projects. The analysis for this project includes a subset of risks from the aforementioned article which apply to this project (Risks Nr. 1 to 6), as well as risks that apply to our team (Risks Nr. 7 to 10), accompanied by a description, the probability of occurrence, the severity, and the mitigation plan:

5.3.1.1 Excessive schedule pressure

- Description: The project could have a schedule overfilled with tasks regarding realistically meeting the milestones when considering the available time and resource budget.
- Probability: Medium
- Severity: High, as this could compromise the project entirely. The goal (and requirement to pass this module) is to produce a functioning system, which could be severely impacted by a shift in the timeline, potentially causing cascading failures for multiple items on the schedule.
- Mitigation: We keep track of and attempt to generously plan the time required for individual milestones. Additionally, the team is trying to accommodate potential delays by working on multiple issues simultaneously.

5.3.1.2 Lack of technical documentation

- Description: The project is not properly organized and documented. While this can be acceptable in a small-scale project, it can become an issue once the projects' complexity exceeds a certain threshold.
- Probability: Medium
- Severity: High, as this will directly influence our ability to produce a proper final design report which is one of the main components of this project. Additionally, the team is delegated to individual responsibilities and relies on documentation as a means of communication.
- Mitigation: Enforce documentation for every step of the way. The team agreed on the strategy that having more documentation is better than less as it retains our ability to delete the information if it becomes irrelevant versus having to reverse-engineer our own decisions to compile a report at the end of the module. We try to use several tools (e.g.: Swagger API automatic documentation) to facilitate this.

5.3.1.3 Lack of a documented project plan

- Description: Without documented project planning, there is a high likelihood that resources such as time and manpower will be mismanaged. This can also cause communication issues regarding deadlines and milestones that must be properly estimated and reached.
- Probability: Medium
- Severity: Medium, as this will reduce the number of features that will be implemented in the available time.
- Mitigation: The team will provide a comprehensive initial project plan and will re-evaluate it as new issues arise or are cancelled. A team member is responsible for ensuring that the planning is up to date.

5.3.1.4 Excessive and secondary innovations

- Description: Overengineering as well not properly researching and implementing existing solutions to problems. In combination with secondary innovations, i.e.: overengineering secondary requirements (which were derived from primary ones) due to creative impulses of the team.
- Probability: High, since we are a motivated group with many ideas and approaches to tackle issues.
- Severity: Medium, since this will reduce the number of features that we will be able to implement if we invest or waste time implementing other features.
- Mitigation: Prioritize the features, discuss possible implementations, and make plans, accordingly, allocating time-budgets for individual functionalities and re-evaluating them accordingly.

5.3.1.5 Requirements creep

- Description: Requirements can and will change over time. It is possible that they are mismanaged and not properly updated and documented. Over time and with scale this can cause multiple issues, such as not being able to properly implement them or implementing redundant or outdate requirements.
- Probability: Medium
- Severity: Low since this project is defined and limited in scope and length the requirements will likely not drastically change.
- Mitigation: The team tries to take requirement changes into account by rapidly developing a prototype with the expectation of changing requirements due to the nature of this project – it will be tailored towards the client's needs and preferences.

5.3.1.6 Lack of scientific methods

- Description: If the project is not developed in small and incremental steps, not properly tested and its evolution not properly documented it can result in confusion and miscommunication.
- Probability: Medium
- Severity: High, since miscommunication and a lack of documentation will impact the difficulty of e.g., integrating the front- and backend during the project.
- Mitigation: The team will adopt a test-driven development process to implement correct functionality on a unit and integration level as well as utilize tools such as Swagger to facilitate documentation and design. Additionally, the team aims to have solid designs before implementing as well as updating them during the implementation process.

5.3.1.7 Inter-team personal conflicts

- Description: Differences in opinion or high stress factors can lead to conflicts between two or more members of the team. This will in turn cause further issues such as not meeting deadlines and diminish communication efficiency in the team.
- Probability: Low
- Severity: Medium – inefficiencies in the workflow will waste time and influence the amount of work that the team can accomplish.
- Mitigation: The team will try to identify personal conflicts and communicate them before they escalate. This way the team can discuss and potentially resolve conflicts before they endanger the progress in the project.

5.3.1.8 Lack of accountability

- Description: If nobody is assigned to a specific role or task then it is possible in turn nobody feels obligated or responsible to complete it. Additionally, sometimes decisions must be made on an individual level instead of waiting for the whole team to vote on an issue.
- Probability: Medium
- Severity: High, because important issues will be forgotten when everybody expects that somebody else will pick up the work. Additionally, if nobody is accountable for a part of the work or functionality, then nobody will make decisions and time will be wasted by involving the whole team in minor decisions.
- Mitigation: Assigning tasks to members of the team on multiple levels. E.g.: One person will be accountable and responsible for organizing the front-end part of the application. This does not mean that they are solely responsible for the work, but they will represent a contact person for the other members of the team concerning an issue.

5.3.1.9 Lack of commitment and differing goals

- Description: Individual members of the team might have different perspectives on the goals they want to achieve in this project. Some might prefer to simply deliver a project that will make them pass the module while others might be interested in exceeding expectations.
- Probability: Medium
- Severity: Medium, since this will not directly compromise the project, but it will limit its potential and could possibly cause personal conflicts or fracture the team.
- Mitigation: The team was assembled prior to this module on the basis that every member has a high level of motivation and expectations for this module. Due to this, the team is on the same page in trying to deliver a project that will reflect the ambition.

5.3.1.10 Absences and equipment malfunctions

- Description: It is always possible that one or more team members are absent, either because of planned but also unforeseen circumstances. Additionally, a disrupted internet connection or broken laptop might cause unavailability of members for a period.
- Probability: Low
- Severity: Low, this will lead to a reduction in available resources for the team overall but should not have a major impact on the overall project
- Mitigation: The team has accounted for this in two ways: First, if a member must be absent, they will inform the team as soon as possible, so that adjustments to the plan can be made. Second, if there are technical malfunctions, the team can provide replacements, or we can meet on campus or at the location of another team member to continue working.

5.4 Planning

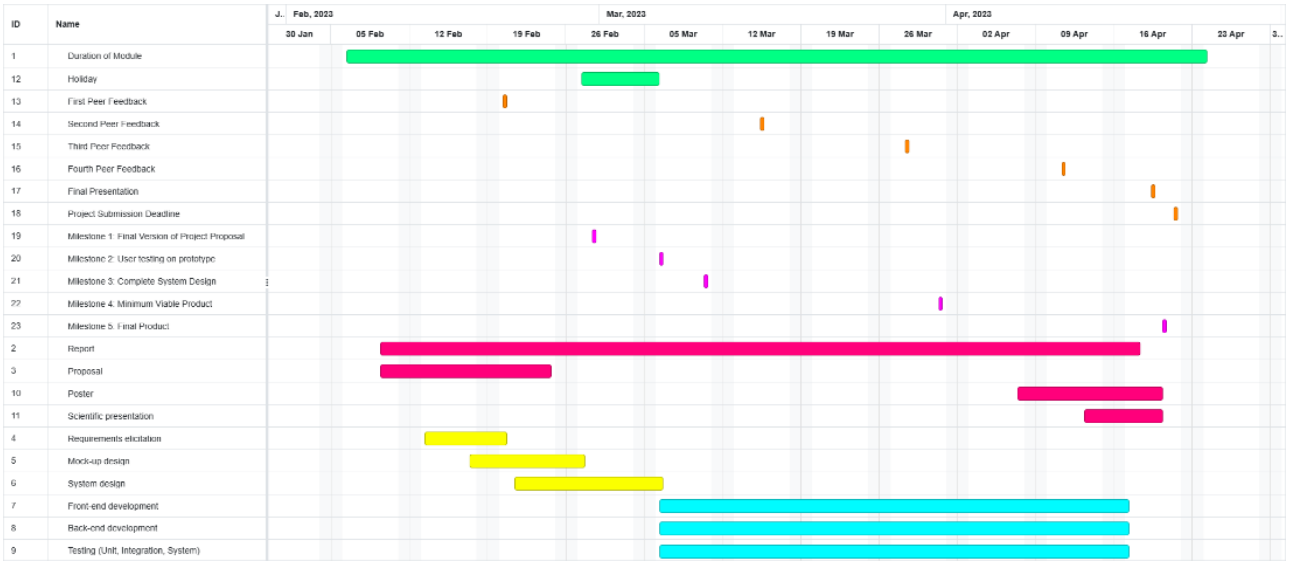


Figure 1: Planning - GANTT Chart

The following image (Figure 1: Planning - GANTT Chart), shows our initial planning for the whole duration of the design project module (A bigger version of this image can be seen in Appendix A: Planning overview). It is divided into several parts.

- University activities (peer sessions, final presentation, and final project submission deadline) are marked in orange.
- Milestones that our team set, internally, for the project are marked in violet.
- Duration of the module and the 1-week holiday are marked in green.
- Writing, presentation, and poster parts for the project are marked in pink.
- Designing work (system and mock-up) is marked in yellow.
- Finally, programming work is marked in blue.

Alongside the planning, a list of 5 milestones was set by the development team:

- Milestone 1: Final Version of Project Proposal – possibly have the final version of the proposal done by 24th February 2023, which incorporates all feedback points from the client, while also satisfying the requirements mentioned in the Design Project Manual, regarding the structure of the proposal.
- Milestone 2: User testing on prototype – perform the final user-testing with a Figma prototype on 06th March 2023, discuss possible additional features to the interface and inconveniences/mistakes in the current design.
- Milestone 3: Complete System Design - Everything related to the designs of systems, showcased by diagrams (class, activity, etc.), API endpoints, sitemaps, mock-ups should be discussed and agreed to by 10th March 2023.
- Milestone 4: Minimum Viable Product – All the must-have requirements should be possibly/preferably implemented by 31st March 2023.
- Milestone 5: Final Product – The final product, incorporating all the must requirements from the client should be delivered by 20th April 2023.

By completing each of them on time the team would have guaranteed a significant progress in delivering a good end product.

Overall, since the creation of the plan in the first weeks of the module, each team member aimed in strictly following the schedule. This helped us in having enough time for the different aspects of the project, namely the implementation, documentation, final-report, poster, presentation, etc. In the next lines, a concise explanation regarding the different milestones will be provided.

The first milestone was achieved on the specified date; however, the client wasn't impressed with the quality of the report. This resulted in prolonging the proposal-writing milestone with one week and doing the activity in parallel with the designing. Nevertheless, the team managed to recover from the slight change in the schedule by producing a significantly better proposal which met the needs of the client and further improved the understanding of all team members regarding the systems.

The second milestone, contrary to the first one was achieved on time. It brought some valuable insight from the client and helped in the implementation of the project.

The third milestone, similar to the second, was completed on time let the team to start the development work during the next week (week 5 from the module).

For the fourth milestone, due to the complexity of the project and many unforeseen circumstances, the development team had the MVP product one week later. This change in the plan wasn't anticipated, however due to the internal organization within the team it didn't influence the overall process and had rather positive consequences.

Last but not least, the fifth milestone was achieved, and the development team had a product, which was in a finished condition and could be sent to the client.

In conclusion, the planning went generally as anticipated. There were some delays and unexpected circumstances, however none of them managed to negatively influence the team in delivering a successful product.

6 System Specification

6.1 Stakeholders

The following table summarizes the list of stakeholders for the Conference Support System

Stakeholder	Details
Client - Overall Steering Committee Chair	The client of the project, one of the responsible people for organizing the ETAPS conference.
Chairs – Committee Chairs	People responsible for scheduling and member selection within a committee (TACAS, ESOP, FASE, etc).
ETAPS	The organizational body of the whole event. It consists of different committees (TACAS, ESOP, FASE)
EasyChair	A third-party conference management system, which is used to upload the conference program for ETAPS

Table 2: Stakeholder Overview

6.2 Requirements

During the initial 3 meetings with the client, a total of 40 requirements were identified. All of the requirements were prioritized according to the MoSCoW analysis (Must, Should, Could, Won't), with 22 "must", 12 "should" and 6 "could" requirements and separated into functional (a total of 30) and non-functional (a total of 10). An extensive table of all the requirements can be observed in Appendix B: Requirements.

6.3 User stories

Requirements might sometimes be unclear to the client and end user since they are written in a technical manner. To facilitate the understanding of non-technical readers, user stories were written for each requirement. In Appendix C: User Stories, all the identified user stories from the first 3 meetings are summarized in tabular form and referenced to the corresponding requirements.

6.4 Acceptance Criteria

In this this section, the development team will provide the acceptance criteria for all the user stories. It will follow the **GWT** format:

- **Given** (some precondition)
- **When** (some action is performed)
- **Then** (what result is expected).

Each individual acceptance criteria will indicate an important requirement (See Appendix B: Requirements).

6.4.1 Acceptance Criteria System 1

User Story: As a Steering Committee chair, I want the system to be able to provide a visual interface which has information regarding the number of available slots per conference. (**RE11**)

Scenario: The Steering Committee chair wants to see the number of available slots per conference from a table.

- **Given** the Steering Committee chair is logged in the system,
- **When** the Steering Committee chair wants to check the visual information overview for the number of available slots per conference,
- **Then** the Steering Committee chair will see a table with the information about participants and the number of available slots per conference.

User Story: As a Steering Committee chair, I want to be able to create a schedule and assign slots of time for multiple parallel conferences. (**RE1**, **RE13**)

Scenario: The Steering Committee chair wants to create a schedule for an upcoming event.

- **Given** the Steering Committee chair is logged in the system,
- **When** clicking on a button “New”,
- **Then** the user will be redirected to a page, responsible for the creation of a schedule**.**

User Story: As an ETAPS Conference speaker/presenter, I want the system to be able to give one presentation to one timeslot at a time across all conferences. (**RE5**)

Scenario: ETAPS Conference speaker/presenter wants to be given one slot at a time across all conferences.

- **Given** a logged in ETAPS Conference speaker/presenter,
- **When** attempting to be assigned to more than one timeslot at a time,
- **Then** the system would indicate that a problem might occur.

User Story: As an Individual ETAPS conference chair member, I want to be able to assign speakers to sessions for my conference. (**RE6**)

Scenario: An individual ETAPS conference chair member wants to have the ability to assign speakers to sessions for his/her own conference.

- **Given** a logged in individual ETAPS conference chair member,
- **When** entering the details of a speaker that will be assigned to the corresponding conference,
- **Then** the speaker will be added to the corresponding conference.

User Story: As a Steering Committee chair, I want to have a clear overview of the program after all the chairs have added their conference information. (**RE8**)

Scenario: The Steering Committee chair wants to have an overview of the program after the whole conference information is added.

- **Given** the Steering Committee chair is logged in,
- **When** clicking on an overview button,
- **Then** an overview of the program with the whole conference information will be displayed.

User Story: As a Steering Committee chair, I want to be able to change the information about the sessions after all the chairs have added their conference information. (**RE7**)

Scenario: The Steering Committee chair wants to be able to change the information about sessions.

- **Given** the Steering Committee chair is logged in,
- **When** clicking on speaker/presenter's table entry,
- **Then** the Steering Committee chair would be able to change the speaker/presenter's data.

User Story: As a Steering Committee chair, I want the system to overview/highlight possible duplicate entries. (**RE9**)

Scenario: The Steering Committee chair wants to see an overview/highlight of possible duplicate entries.

- **Given** the Steering Committee chair is logged in,
- **When** looking at an overview of the conference schedule,
- **Then** the system should overview/highlight the possible duplicate entries**.**

User Story: As a Steering Committee chair, I want to be able to create four different types of timeslots: Presentations, Miscellaneous, Plenary Sessions, Breaks (**RE3**)

Scenario: The Steering Committee chair is creating the schedule

- **Given** the Steering Committee chair is logged in,
- **When** creating the conference schedule,
- **Then** they should be able to assign four different types of timeslots: Presentations, Miscellaneous, Plenary Sessions, Breaks

User Story: As a Steering Committee chair, I want to be able to clearly see (with the help of highlighting) when people are present in multiple conferences simultaneously (duplicates). (**RE9**)

Scenario: The Steering Committee chair wants to have an aid from the system for spotting people who might be present in multiple conferences simultaneously.

- **Given** the Steering Committee chair is logged in,
- **When** looking at an overview of the conference schedule,
- **Then** the system should mark all people who might be assigned in multiple conferences simultaneously.

User Story: As a Steering Committee chair, I want to generate the conference program in the YAML format, which is usable for the ETAPS website to display the program. (**RE10**)

Scenario: The Steering Committee chair requests that the conference programme to be generated in YAML format.

- **Given** The Steering Committee chair is logged in,
- **When** wanting to generate the conference program in YAML format,
- **Then** by clicking on a button "Generate", the conference programme will be exported in YAML format.

User Story: As an Individual ETAPS conference chair member, I want to be able to specify that a presentation can last from 15 to 120 minutes. (**RE2**)

Scenario: An Individual ETAPS conference chair member enters a duration for presentation from 15 to 120 minutes.

- **Given** a logged in Individual ETAPS conference chair member,
- **When** adding a speaker/presented to the programme,
- **Then** an option for the duration from 15 to 120 minutes should be provided.

6.4.2 Acceptance Criteria System 2

User Story: As a chair member of one of the ETAPS conferences, I want to be able to enter the data in a form on the website. (**RE14**)

Scenario: A chair member of one of the ETAPS conferences wants to enter data in a website form.

- **Given** a logged in chair member of one of the ETAPS conferences,
- **When** planning to add a new person to the conference committee,
- **Then** a form should be provided, enabling the chair member to enter the personal details of the new potential committee member.

User Story: As a Steering Committee chair, I want to identify whether steering committee members are assigned to multiple committees simultaneously. (**RE22**)

Scenario: A Steering Committee chair wants to identify whether steering committee members are assigned to multiple committees simultaneously.

- **Given** a logged in overall Steering Committee Chair member,
- **When** finalizing the overall multi-conference schedule for ETAPS,
- **Then** an overview should be provided, indicating whether people are assigned to two different program committees, even if their names are spelled slightly differently.

User Story: As a Steering Committee chair, I want the system to output an overview/highlight of possible inconsistent inputs in the names between the committees four conferences. (**RE22**)

Scenario: A Steering Committee chair wants the system to output an overview/highlight of possible inconsistent inputs in the names.

- **Given** a logged in overall Steering Committee Chair member,
- **When** finalizing the overall multi-conference schedule for ETAPS,
- **Then** an overview should be provided, indicating if two people have the same contact details but their names are spelled differently, are differently ordered, or mistyped.

User Story: As the Easychair organization, I want to receive the list of people in a specific format, so that I can send the invitations automatically. (**RE17**)

Scenario: The Easychair organization wants to receive the list of people in a specific format (“FirstName” “LastName” <EmailAddress>).

- **Given** the Easychair website, for managing invitations to the ETAPS conference,
- **When** wanting to invite a list of people,

- **Then** said list can be uploaded, one person per line in the format: “FirstName” “LastName” <EmailAddress> (including ‘<’ and ‘>’ and ‘”’).

User Story: As the steering committee chair, I want to be able to indicate whether potential program committee members have accepted or rejected the invite from the Easychair website. (**RE18**)

Scenario: A Steering Committee chair want to indicate whether a potential program committee member has accepted or rejected the invite.

- **Given** a logged in overall Steering Committee Chair member,
- **When** people have been invited using the Easychair website and either rejected or accepted the invitation,
- **Then** the overview page of the planned program committees will provide the option to display and edit the status of the individual members with at least the values: ‘Accepted’, ‘Rejected’, and ‘Pending’.

6.4.3 Acceptance Criteria Combined Systems

User Story: As a Steering Committee chair, I want to be able to interact with the data visually. (**RE28**)

Scenario: A Steering Committee chair want view and edit conference schedules via a GUI (graphical user interface).

- **Given** a logged in overall Steering Committee chair member,
- **When** visiting the conference support system website,
- **Then** they can view and edit the conference schedules and committees utilizing a graphical user interface.

User Story: As an individual conference chair member, I want to be able to interact with the data visually. (**RE28**)

Scenario: An individual conference chair member wants to edit and view the conference schedules and committees.

- **Given** a logged in Conference chair member,
- **When** visiting the conference support system website,
- **Then** they can view and edit the conference schedules and committees utilizing a graphical user interface.

User Story: As a Steering Committee chair of the ETAPS conferences, I want only people with permission to have access to the data on the website. (**RE26**)

Scenario: A Steering Committee chair of the ETAPS conferences, wants only authorized people to access the data on the website.

- **Given** an unauthorized user that is not part of any of the conference chairs,
- **When** trying to access the conference support system website,
- **Then** they will be prevented from viewing or editing the data.

User Story: As a Steering Committee chair of the ETAPS conferences, I want the system to be able to impose people on entering information correctly. (**RE30**)

Scenario: A Steering Committee chair of the ETAPS conferences, wants the system to impose a uniform format for entering information.

- **Given an** overall Steering Committee chair member and individual conference chair members,
- **When** attempting to organize the schedules and committees of the individual conferences by contacting the individual conference chair members,
- **Then** the information provided in the response should be uniform and have a standardized format.

User Story: As a Steering Committee chair, I want to conveniently enter the data in templates with smart default selections. (**RE29**)

Scenario: A Steering Committee chair wants to have default selections and data templates.

- **Given** a logged in Steering committee chair member,
- **When** initially creating a new conference schedule or committee for the program chair members to fill with data,
- **Then** fields that have to be entered or selected should provide default values that are likely to be selected or have been selected in previous years.

User Story: As a Steering Committee chair, I want to modify data templates if they need to be adapted. (**RE41**)

Scenario: A Steering Committee chair wants to edit data template.

- **Given** a logged in overall Steering Committee Chair member,
- **When** an initial template has been created for a multi-conference schedule or committee,
- **Then** this template will be editable at any time with the changes reflecting for all users.

User Story: As an individual ETAPS conference chair member, I want to conveniently enter the data in templates with smart default selections. (**RE29**)

Scenario: An individual ETAPS conference chair member wants to conveniently enter data in templates.

- **Given** an individual ETAPS conference chair member,
- **When** having received a template for a conference schedule or committee that must be filled in for this years' conference,
- **Then** it will be avoided that names have to be entered more than once and once the program committee has been filled the rest of the members will be placed on a reserve list and will be moved from the reserve list automatically if an invited committee member has rejected the invitation.

User Story: As a Steering Committee Chair, I want to be able to access the website with my Firefox Browser and on my laptop or desktop computer. (**RE33, RE35**)

Scenario: A Steering Committee Chair wants to open the website from Firefox.

- **Given** the overall Steering Committee Chair member,
- **When** trying to access the conference support system website using their laptop or desktop computer,
- **Then** it will be possible to access the website and all its functionalities with the Firefox browser.

User Story: As an individual ETAPS conference chair member, I want to be able to access the website with my Firefox browser from my laptop or desktop computer. (**RE33, RE35**)

Scenario: An individual ETAPS conference chair member wants to open the website from Firefox.

- **Given** an individual ETAPS conference chair member,
- **When** trying to access the conference support system website using their laptop or desktop computer,
- **Then** it will be possible to access the website and all its functionalities with the Firefox browser.

While these acceptance criteria are not covering all of the requirements, they are deemed essential and cover every *must* requirement. They will be used in the testing stage (once implemented) and represent essential functionalities of the system. These acceptance criteria helped us negotiate an “acceptable” state of the system with the client after eliciting initial requirements and have been included in the initial project proposal. They have been adapted with the modification of individual requirements over the period of the project.

6.5 Risk Analysis

There are some potential risks when using our developed system. We analysed the probability and severity of them, and classified the risk levels into low, medium, and high. We also developed strategies to mitigate or avoid these risks.

6.5.1.1 Inconsistent input format

- Description: This risk relates to the difficulty in ensuring that all users consistently provide the required information in the correct format.
- Probability: High. Users are likely to provide inconsistent information, such as differences in capitalization, variations in the order of their given name and family name, or the use of different delimiters (e.g., a comma or a space) to separate the names.
- Severity: High. This could result in scheduling conflicts and affect the accuracy and reliability of the conference information.
- Mitigation: The system should have clear indicators and fields for users to correctly enter their information and automatically parse and check for input data.

6.5.1.2 People scheduled for multiple sessions simultaneously.

- Description: This risk relates to the possibility of scheduling multiple sessions at the same time.
- Probability: Medium. This may be due to input inconsistencies. However, this risk can be overseen by the OSCC.
- Severity: High. This could result in scheduling conflicts and potential frustration for conference participants.
- Mitigation: The system should have a mechanism that can prevent duplicated sessions and indicate to the OSCC when scheduling conflicts occur.

6.5.1.3 Privacy concerns

- Description: This risk relates to the protection of personal and confidential information, such as contact information that is entered into the system.
- Probability: Medium. Participants' personal information, such as names, emails, could be shared without their consent.
- Severity: High. Participants' personal information may be stolen or misused.
- Mitigation: The system should limit the amount of personal information collected necessary for conference management and obtain explicit participants' consent before collecting, storing, and using the information. Additionally, the system will regulate the data in a secure manner.

6.5.1.4 User adoption

- Description: This risk relates to the difficulty in getting ETAPS users to use the system as intended.
- Probability: Medium. If users are unfamiliar with using our developed system or find it difficult to use, they may be less likely to adopt it.
- Severity: Low. The usability of our developed system could result in suboptimal user experience. However, it would not cause great issues with the conference management.
- Mitigation: The system should be designed to be user-friendly and "human-editable" to both the ETAPS CC members and OSCC – additionally a user guide will be provided.

7 Process

In this section, the overall process through the module will be explained. This will include utilized tools, methodology of development, communication with client and other relevant aspects that influenced the outcome of product.

7.1 Tools & Existing Literature

For our stack, we use ReactJS for front-end, FastAPI for back-end and MongoDB as a database for storing the information (as depicted in Figure 2: Technology Stack).

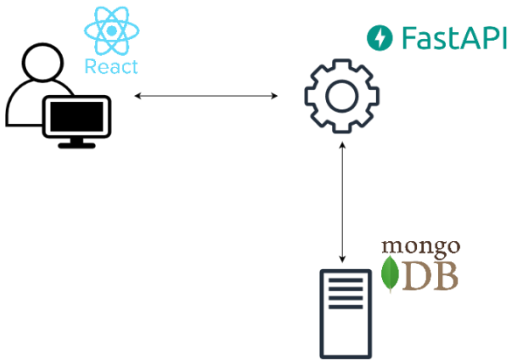


Figure 2: Technology Stack

7.1.1 Front-end

The front-end is an essential part of every web-based system. The main responsibilities of it include defining a robust interface structure which is adaptable to changes, providing a good balance between functionality and aesthetic design, visual aid for the users. We decided to use a framework for it because it provides us with pre-written components to use in the development of the website. Therefore, choosing an appropriate front-end solution is of crucial importance. Currently, there exist 3 widely used and dominant frameworks for front-end, namely Angular, React and Vue.js and each of them has its own strengths/limitations.

Angular is the second most popular front-end framework in the world. It was developed by Google to solve their problems while building front-end solutions. The most notable advantages of Angular include high performance, Angular Material (a design product which has in-built icons, ready solutions for popular components that could be seen on webpages, such as cards, input boxes, forms, buttons, etc.), support for components which isolate parts of the system that could be reused in multiple places and two-way data binding (way to share data between Angular components) that facilitates communication with backend. Regarding the limitations, the framework has a steep learning curve. It is mostly appropriate for larger projects which require a variety of different interactions with the backend. Moreover, certain aspects of Angular are made significantly difficult, compared to its competitors, due to the design choices of the framework architects. Therefore, after weighing the advantages and disadvantages of utilizing Angular, our team decided to avoid using it, given the scope of our problem and project.

Next on the list is Vue.js. It is the third most popular front-end framework in the world. Some of the biggest advantages Vue.js offers are high performance, easy third-party integration and it is the easiest to learn, compared to Angular and React. Regarding the limitations, Vue.js is not

appropriate for larger projects due to the lack of scalability, has difficulties with two-way binding and limited technical capabilities. Considering the pros and cons of Vue.js, our team agreed on not using it, since it would offer limited possibilities/functionalities and flexibility to our project.

React is the most popular front-end framework in the world. It could be utilized to create an animated and interactive user-interface for the web application. The primary reasons for us choosing React (possibly in conjunction with MaterialUI which is like Angular Material) are due to it having a very strong community support, being easy to learn and use with multiple capabilities provided withing the framework. Furthermore, the reusable custom components (like Angular) and the modular structure of the framework will enable the team to design an efficient application and re-use elements (components) in various places. While parts of the team have experience in front-end programming outside of the Technical Computer Science program, there is little experience with using React. The strong community support and popularity will provide access to a lot of resources, documentation, and literature. This will enable the team to familiarize themselves with the framework. Due to the aforementioned reasons, React was decided to be our front-end tool, alongside MaterialUI.

7.1.2 Back-end

FastAPI is a Python based framework and will be utilized to develop the RESTful API in the back-end of the web application. In contrast to ReactJS it is a relatively young framework, being released in the year 2018. Regardless of that it has gained immense popularity for its healthy balance of built-in and underlying features which will be highly applicable for the web application we intend to create. The first feature that stands out is that it is programmed in Python. The flexibility of the Python language, especially its notoriety for processing data will allow the team to extend the functionalities of the back-end to facilitate exactly these requirements. Also, the framework supports concurrency, exception handling and data validation by default, facilitating a correct and fast interaction between the front- and back-end. Since FastAPI is built upon the OpenAPI standard, the API is self-documenting. This will enable us to properly document, test and communicate the API endpoints between the front- and back-end teams and allows for a faster and less error-prone application development. Ultimately, the framework was chosen by the team because multiple members have experience with the Django and Flask frameworks. The primary argument to forego the two other alternatives is that Flask is extremely lightweight and has barely any functionalities built-in by default (outside of the RESTful framework), while Django in contrast has too many base functionalities that are not necessarily needed. Therefore, due to the aforementioned reasons, FastAPI was decided to be the main back-end framework.

7.1.3 Database

In the context of databases, there exist 2 types which dominate the market, namely SQL and NoSQL. The first option offers relational databases, meaning the information is organized into different tables with connections from one table to another one. The main advantages of SQL databases (e.g., PostgreSQL, Oracle, MySQL) include concrete guidelines on how to design relational database structure, possibility to join data from 2 tables, efficiency for processing queries, being the standard for more than 30 years. On the other hand, NoSQL databases (MongoDB, Cassandra, Neo4j) provide flexible data models, horizontal scaling [3], ease of maintenance and have a shallow learning curve, compared to relational databases. After

considering the benefits of both SQL and NoSQL databases, our team decided to apt for the NoSQL option and more specifically the MongoDB document-based database. Currently, it is one of the most used databases in the world and continually growing in terms of popularity, with a significant market share. Additionally, together with FastAPI and React, the three of them form the FARM stack [4], which would additionally facilitate our development and give us the time and opportunity to focus on other important tasks related to the project. Thus, for the database, the team decided to use the non-relational database MongoDB.

7.1.4 Figma

Designing a website interface is a difficult task. There are a lot of aspects that should be taken into consideration, such as button layout, background colours, etc. To achieve that, our team decided to use a tool for this, and next phase called Figma [5]. It is a collaborative, cloud-based app, dedicated to easily designing interfaces [6]. Thus, it has all the necessary functionalities that the development team needed.

7.1.5 Git & GitLab

Version control is one of the most important aspects of software development. It aids developers in managing the changes in files [7], which ensures a smoother collaboration process. Even though there exist multiple version control tools on the market, one of them is the standard and used by over 90% of the users [8], namely Git. The idea behind using this tool is not only due to the popularity of it, but also since all the development team members have experience with it.

Git is the version control system. It tracks which files were changed and offers various features such as “commit”, “post”, “pull”, etc. However, they work on a local basis. To collaborate remotely with the other members, a version control self-hosted management tool should be used. Thus, the development team chose GitLab. The main idea behind deciding to use GitLab is due to it being user-friendly and having previous experience with the tool.

7.1.6 Discord

Communication is key for the success of every project; thus, it is important that all the team members have a regular communication not only with the client, but also within the team. On the market, there exist a lot of different solutions, such as conference tools (e.g., ZOOM, Microsoft Teams), Voice over IP messengers (WhatsApp, Skype), nevertheless all of them are limited to a certain number of features. For the development team, it was important to have an app which not only has messaging features, but also video and call capabilities. After looking at all the possibilities, the team decided that the best option for such tasks is Discord. It not only offers all the aforementioned features but is also easy to use. Through the app, multiple channels could be created where specific matters are discussed (planning, testing, report-writing), so that the team has each topic organized. Overall, Discord was the best possible option for the team’s needs.

7.1.7 Trello

Organizing and categorizing lists of tasks could be a challenging process. In the initial stages of software development, once the designing and requirements elicitation steps take place, an overwhelming number of objectives is set and sometimes it is difficult to organize all of them. Kanban [9] is a popular framework which address the abovementioned challenges in a systematic way. It displays all the tasks that the team must complete on a board with different columns (to do, in progress, etc.). One of the most widely accepted tools which utilizes the Kanban framework is Trello. It is a simple table-based tool which could be used for all the aforementioned tasks.

7.1.8 SharePoint

Most software development projects, through the different stages, require the writing of numerous documents. Therefore, it is important to organize all these files in clear structure, ensure that all of them could be easily accessed and edited from other team members and facilitate the overall collaboration process. SharePoint is a Microsoft tools which enables all the specified functionalities and features. It enables multiple users editing the same file and has a clear structure. Finally, it is connected to the University of Twente university accounts which further facilitates the whole process.

7.2 Methodology

Agile is one of the most common software methodologies [10]. It concentrates on an “iterative approach”, which prioritizes on creating something workable, also known as Minimum Valuable Product (MVP), compared to Waterfall, where the focus is on following concrete steps while developing a software product. Therefore, Agile development is significantly more flexible, compared to other methodologies and the development team is responsive to changes in the project.

On the other hand, Waterfall is the complete opposite. It has dedicated steps, which should be taken consecutively, without the possibility to switch something. These steps can be divided into “Requirements”, “Design”, “Implementation”, “Verification” and “Maintenance” [11]. While the ordering of the phases seems relevant and understandable, it is not always the case that software projects go exactly as planned initially. Thus, if a Waterfall methodology is followed, it is impossible to include additional requirements or suggest that something should be changed in a phase different than “Requirements”, due to the consequential approach. This is a big limitation and is one of the reasons for creating the Agile manifesto.

After the short introduction of both methodologies and identifying the advantages and disadvantages of them, it is time to share our approach for the project. It is hybrid between Agile and Waterfall. The whole 10-week duration of the module is separated in 4 parts, namely “planning”, “requirements elicitation & system design”, “implementation & testing” and “presentation”. The intention was to have a structure which is similar to the Waterfall model, so that it is clear what should be done for certain periods of time, and to utilize part of the Agile advantages, such as the flexibility towards changing requirements and the iterative approach.

Overall, with the hybrid methodology our team managed to achieve some excellent results. It helped us in meeting our client’s needs and implementing some additionally requested features rather easily.

7.3 Communication

To deliver a product that meets the expectations of the client, it is important to communicate regularly not only within the team, but also with the client. Therefore, the organization in terms of communication was agreed during the first weeks of the module and was followed throughout the whole module.

With the client, our team had regular weekly meetings on Tuesday at 17:00, most of the time. During these sessions, the weekly progress was discussed alongside possible additional requirements, confirmation regarding certain choices and more. These meetings were crucial for the successful end of the project and served important role in establishing problems early in time.

During the first three weeks (the primary requirements elicitation phase), reports were compiled and send to the client after every meeting to verify that we understood the contents of the meeting and avoid miscommunication.

Moving to the communication inside the development team, it was established that all the members will meet on daily basis (Monday to Friday) at 10:00 to discuss the current progress and possible setbacks that might arise. During the whole module, our team was in constant communication and each member was present during the daily meetings. This not only served as a natural teambuilding activity and let us understand each other well, but also mitigated the risk miscommunication and mismanagement of project-related activities. Overall, our team had high spirit throughout the whole module and now works as a well-oiled machine.

8 Prototyping

After eliciting the requirements from the first 3 meetings with our client and writing a list of user stories, the development team had almost all the information necessary to propose sample systems to the client. To guarantee a smooth process, the development team followed one of the most popular methodologies for designing an interface for a system, namely separating the interface designing task into 3 phases, wireframe, mock-up, and prototype (source 1). In the following subsections, each of the phases will be explained in more detail, together with argumentation why it is important.

8.1 Wireframe

The first phase is the wireframe. The main idea behind it is to show the client the general structure of the different pages (where certain buttons will be located, will the page have a navigation bar, or maybe sidebar, etc.). Most of the time the wireframe is something which is sketched on paper and requires small amount of effort and time. Moreover, the wireframe could be sent quickly to the client inquiring whether the development team correctly understood the main objectives and tasks that each page should support. In case the wireframe is satisfying the needs of the client, it could be directly proceeded with the next phase. Otherwise, if it doesn't meet the expectations of the client, then a new wireframe could be easily sketched and sent again for confirmation. Therefore, it was crucial to follow the 3-phase methodology for many reasons, the most notable one being in case of a misunderstanding between the client and the development team, the earlier it was identified, significantly less time and effort would have been wasted.

8.2 Mock-up

After confirming the general structure of each page with the client using the wireframe, the team started to work on the mock-up. The main intention in this phase of designing is to produce a high-fidelity render of your design that showcases how the finished product would have possibly appeared. Therefore, the mock-up phase improved upon the wireframe, showing design of buttons, choice of background colours, logos, navigation bars, sidebars, etc. It was important to invest time and effort into delivering a mock-up that would be representative to the final product. This not only helped during the final phase of designing, but also served as a blueprint for the front-end developers while creating the solution and was shown to the client to verify the expectations during one of the meetings.

8.3 Prototype

The final and concluding phase of the design methodology is the prototype. It includes all the work completed in the previous stages and has the main objective of delivering an interactive mock-up, called prototype. Most of the time, in terms of visual appearance, the mock-up and the prototype are almost identical. The prototype was used during the user testing meeting, where the client, while being disposed to a limited number of functionalities, interacted with the system, and attempted to finish a set of tasks assigned from the development team. Therefore, the main advantage of this phase was performing the user testing, where different stakeholders could share their opinion regarding the points of improvement, before starting the actual implementation work. During the last phase, it was still relatively easy to incorporate most of the suggested changes

without requiring serious effort. Figure 3 shows the iterative process of starting off with the Wireframe and finishing with the Prototype.

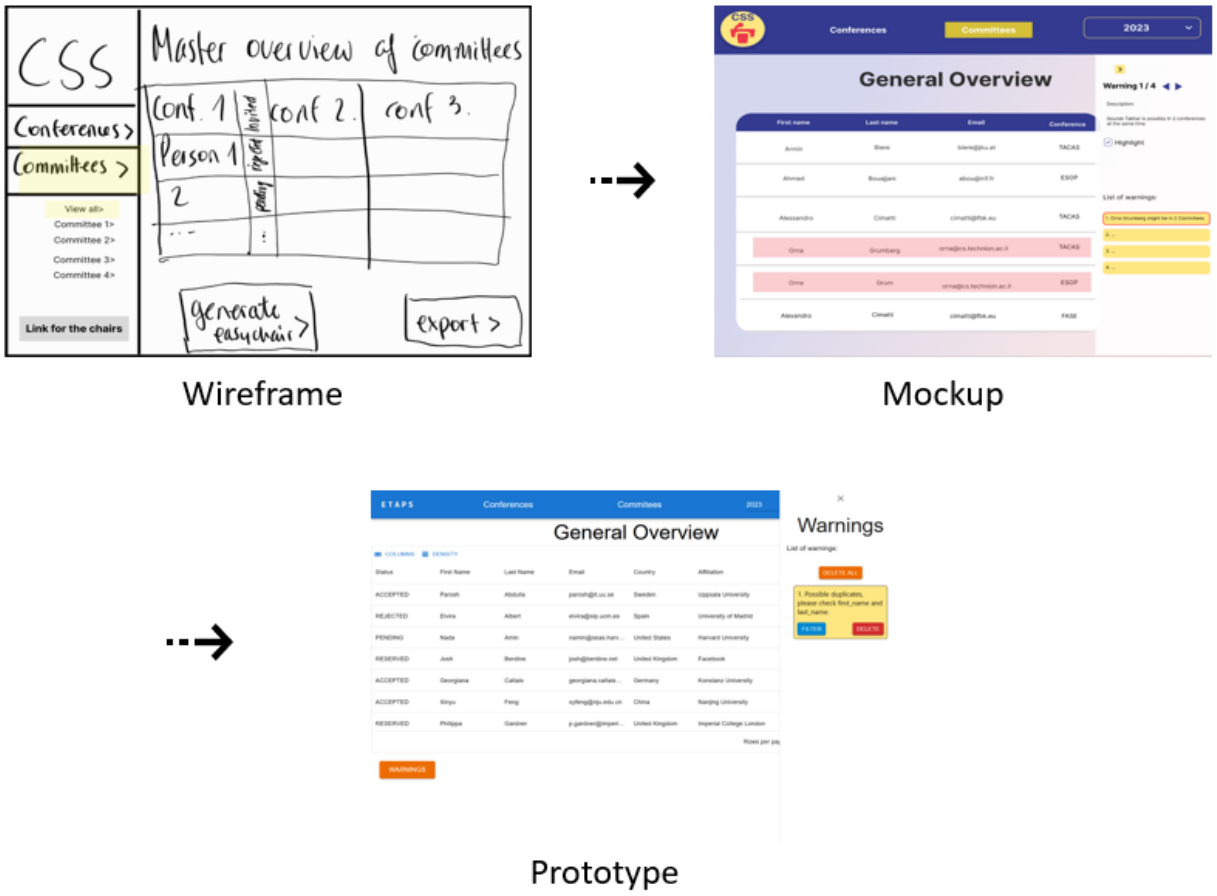


Figure 3: From Wireframe to Prototype

8.4 Proposed Interface

In the next subsections, an explanation of the various functionalities and aspects for the different Figma pages will be provided. This could serve as an information regarding the team understanding of the system between the 2nd and 4th week of the module, based on the elicited requirements. However, it is important to emphasize that not everything in the interface design was implemented the exact same way for the final product. Some of the pages were removed, others were added, there were changes in the functionality.

8.4.1 Landing page

The following page (Figure 4: Landing Page) was designed with the intention of enabling committee chairs to choose their committee from the list of created committees by the admin of the system (the Overall Steering Committee chair of the ETAPS conferences). With the proposed design, a chair of TACAS would be able to add information regarding TACAS conference and check the details for the rest of the committees.

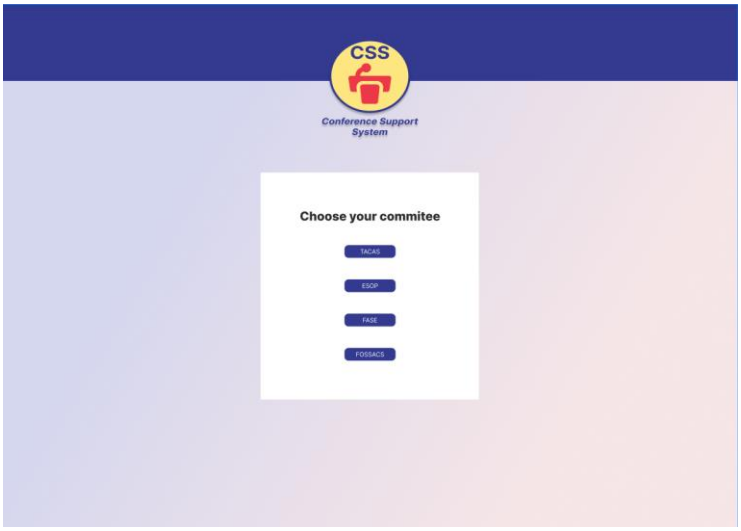


Figure 4: Landing page

8.4.2 Choosing a session

In this page (Figure 5: Session Page) all the program information per committee would be available. The data would have been split into multiple blocks, each dedicated per specific day and every day having cards assigned to topics.

Figure 5: Session page

8.4.3 Assigning slots

By clicking on a topic on the session page (Figure 5: Session Page) the user would be redirected to the assigning slots page (Figure 6: Presentation slot assignment). The main possibilities would be to add presentation for the topic, either by typing it, or by using the uploading of accepted papers functionality.

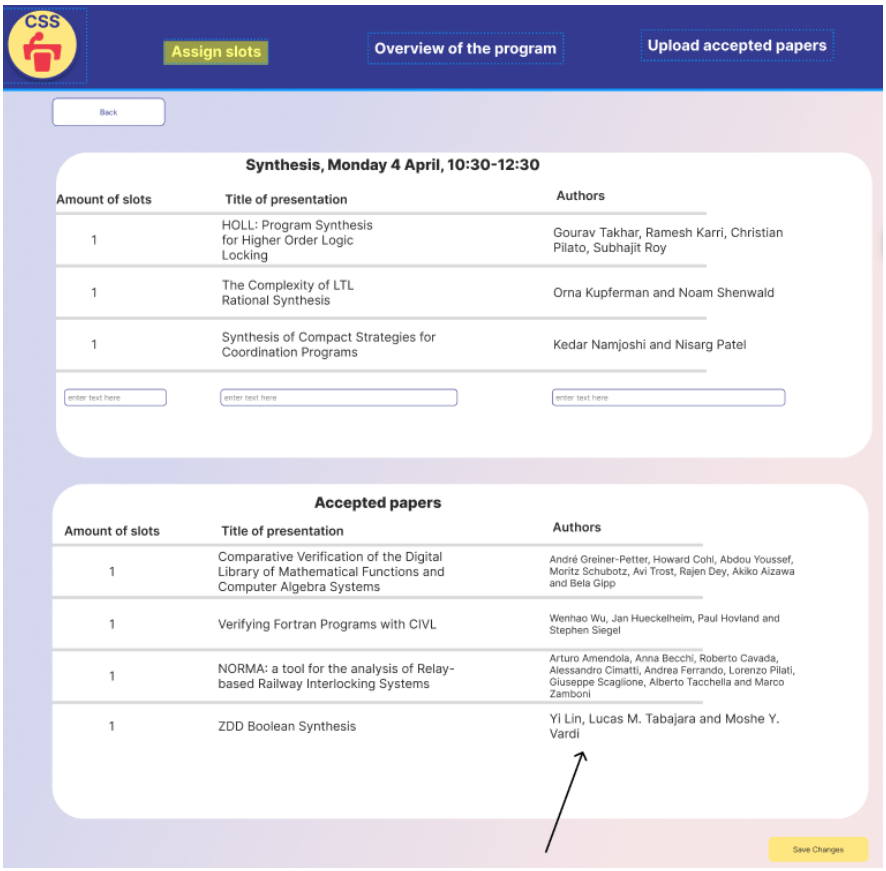


Figure 6: Presentation slot assignment

8.4.4 Overview

The overview page (Figure 7: Overview of presentations page) would offer all the topic information, including presentations, for each day of the conference per committee. This would enable committee chairs to check the general organization and possibly spot some mistakes that could be fixed.

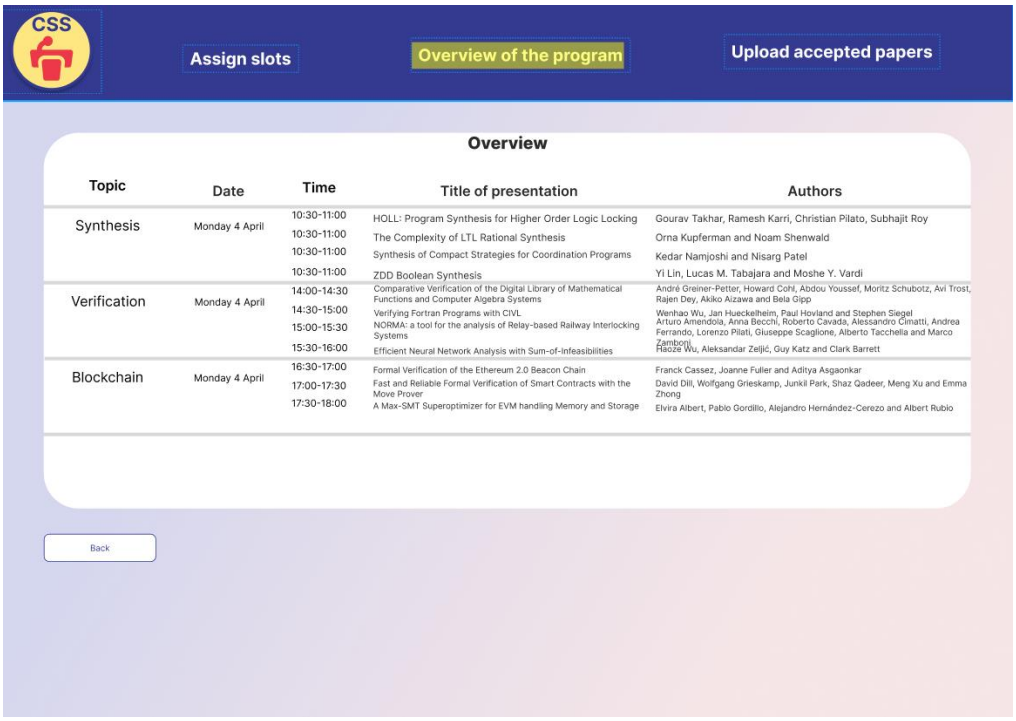


Figure 7: Overview of presentations page

8.4.5 Committee page

The committee page (Figure 7: Overview of presentations page) was supposed to serve as a detailed overview of all the committee members per conference who have intentions of participating in the conference and showing their work. This page was mainly designed for the committee chair, who would be able to add members (Figure 10: Add new member) to the corresponding conference with different status, namely **accepted** (green check mark), **pending** (grey minus symbol) and **rejected** (red minus symbol). Furthermore, there was the reserved and primary list which were dedicated to separate applicants into 2 categories. People who would be in the primary list and accept the invitation, thus participate in the conference, whereas people in the reserved list would participate only if someone from the primary one rejected the offer. Regarding the table functionalities, the committee chair would be able to edit the data of participants, delete and add columns (Figure 9: Add new column).

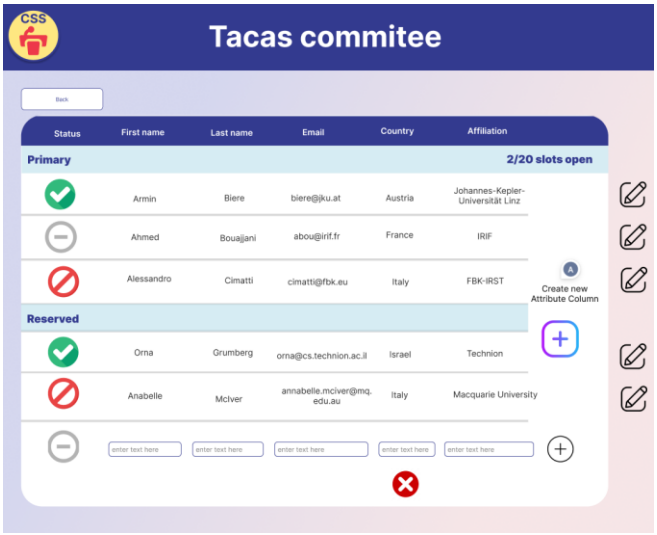


Figure 8: Committee page

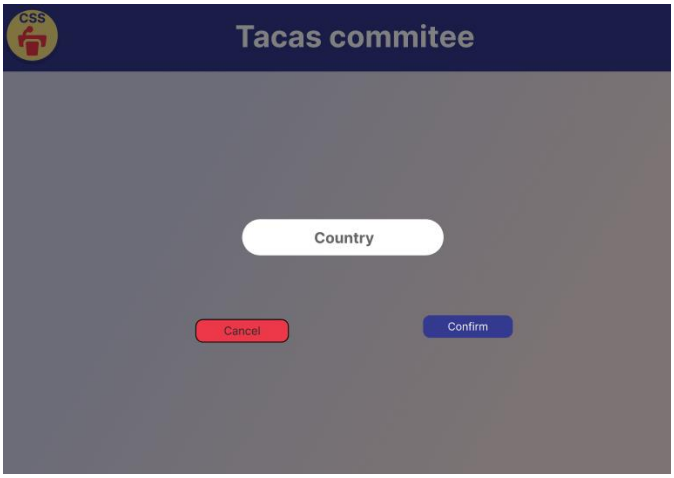


Figure 9: Add new column.

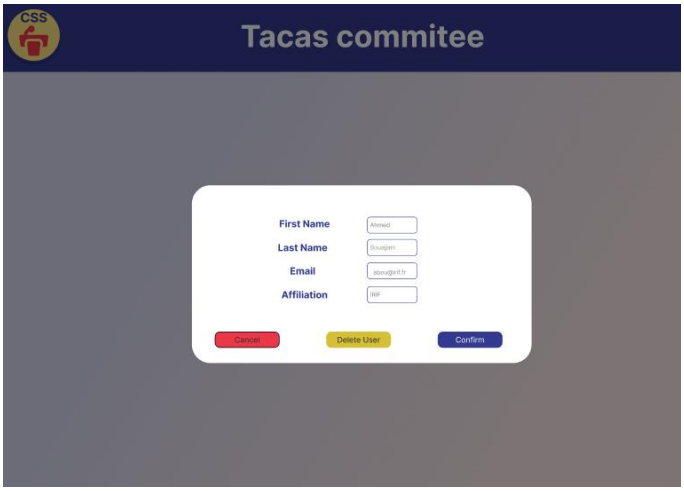


Figure 10: Add new member.

8.4.6 Landing page

The following page (Figure 11: Landing page) was designed to be the first encounter of the admin user (the Overall Steering Committee chair of the ETAPS conferences), where specific credentials should be provided, in order to proceed to the next page.

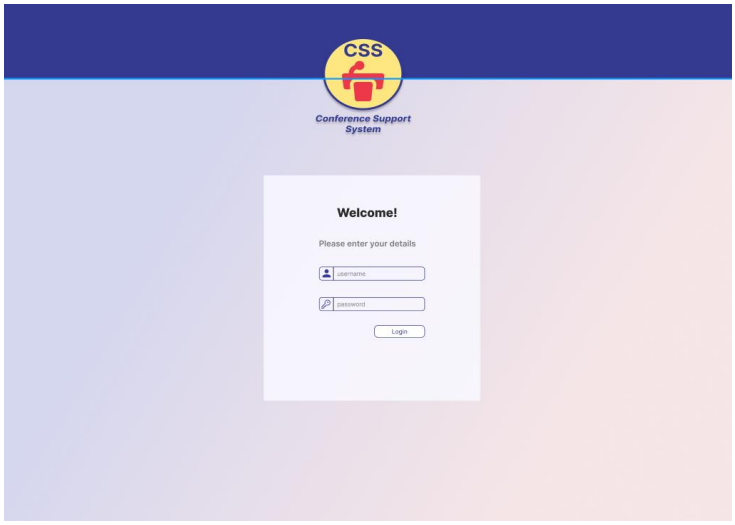


Figure 11: Landing page

8.4.7 Main page

After successfully entering the credentials, the admin user would be able to choose either the Conference Scheduling System or the Committee Seat System (Figure 12: Main page). For each of them, an option would be provided to proceed with creating something new for either of the systems or loading existing data. Furthermore, in the navigation bar, the admin user would be able to easily switch between the 2 systems and change the year for which the actions are performed.

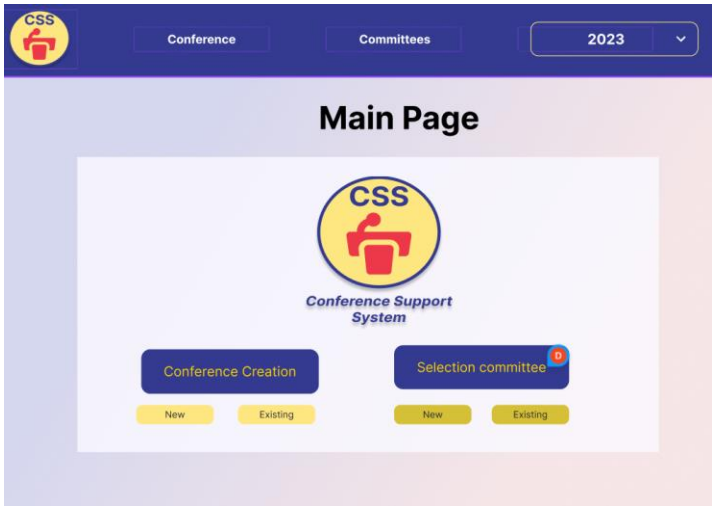


Figure 12: Main page

8.4.8 Creating new committee

Given that the “new” option for selection committee is clicked, a process for creating a new committee would be started (Figure 13: Creating new committee). Then, the admin user would have the option of choosing the year for which the committee selection process will be created, the conference and the table columns (name, year, contact or others).

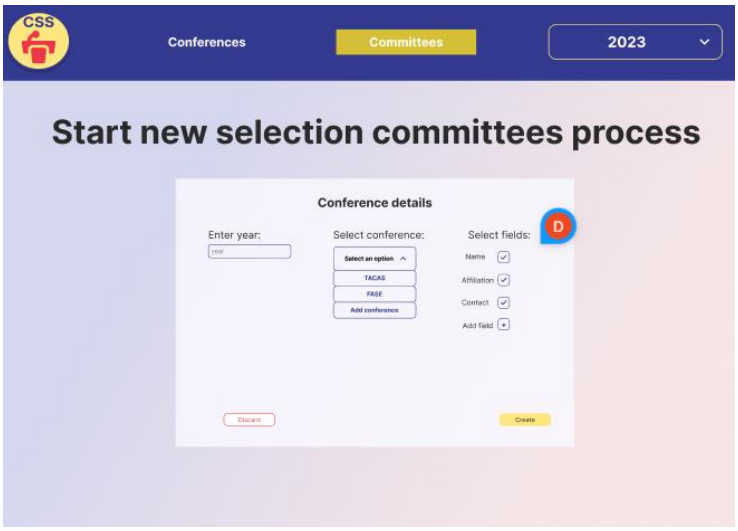


Figure 13: Creating a new committee.

8.4.9 General Overview of the Committee members

On this page (Figure 14: General overview), a table with all the members would be displayed with a “warnings” button, dedicated for highlighting the possible duplicates that might be in the table (Figure 15: General overview duplicate check), so that the admin user could check whether the warnings are relevant and edit the data if necessary.

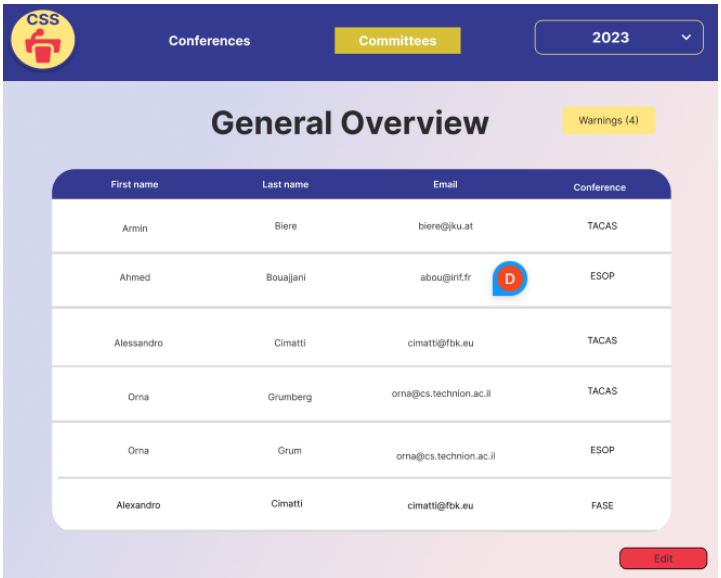


Figure 14: General overview

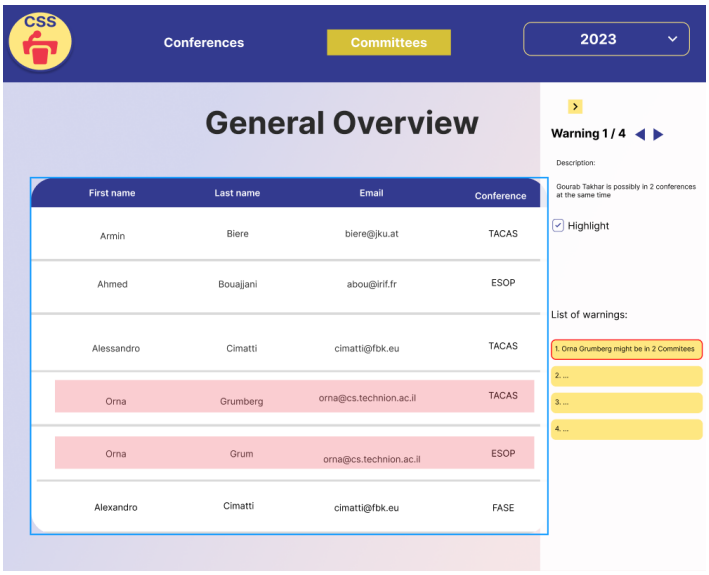


Figure 15: General overview duplicate check

8.4.10 Creating a schedule

In case the “new” box for Conference creation is clicked, then the user would be redirected to the creating a schedule page (Figure 16: Creating a schedule). Here, the possibilities are 2, choosing to create a schedule from scratch, which would require adding all the conference information regarding the different types of sessions (plenary, breaks, etc.). On the other hand, the option from template would provide the user with schedules from previous years which could be loaded for the desired year (most probably the current one) and edited according to the needs of the client.

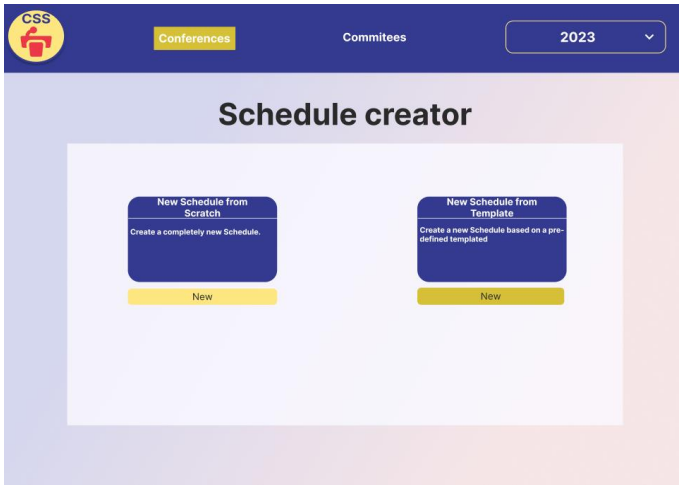


Figure 16: Creating a schedule.

8.4.11 Existing schedule

When the “existing” option from the main page (Figure 12: Main page) is selected, a table with the whole schedule for all the committees per day will be displayed, where the admin user would be able to edit the time, title of presentation and authors’ field (Figure 17: Detailed overview). Furthermore, similarly to (Figure 14: General overview), a highlighting functionality would be provided for the possible duplicates in the authors’ field (Figure 18: Detailed overview duplicate check), so that the user could see if there exists a conflict in the schedule (a person assigned to two presentations at the same time, for instance).

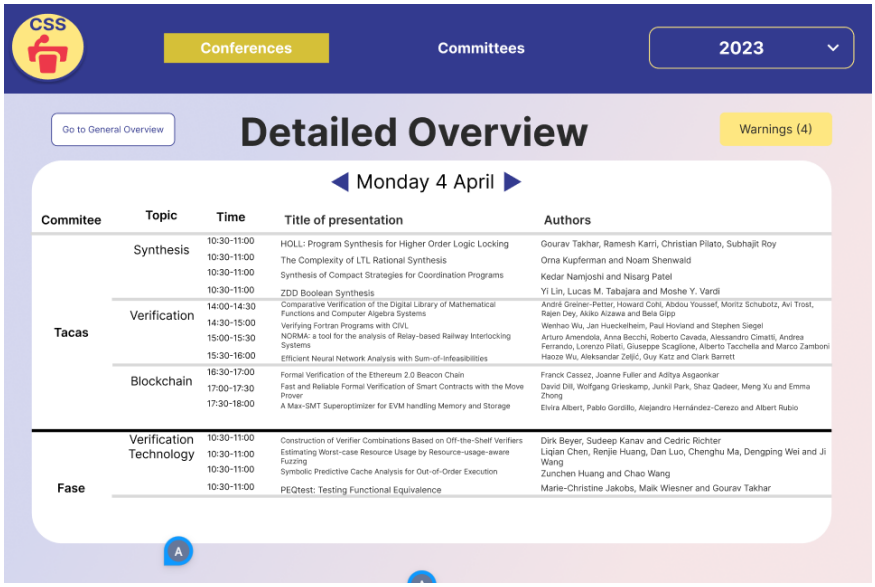


Figure 17: Detailed overview

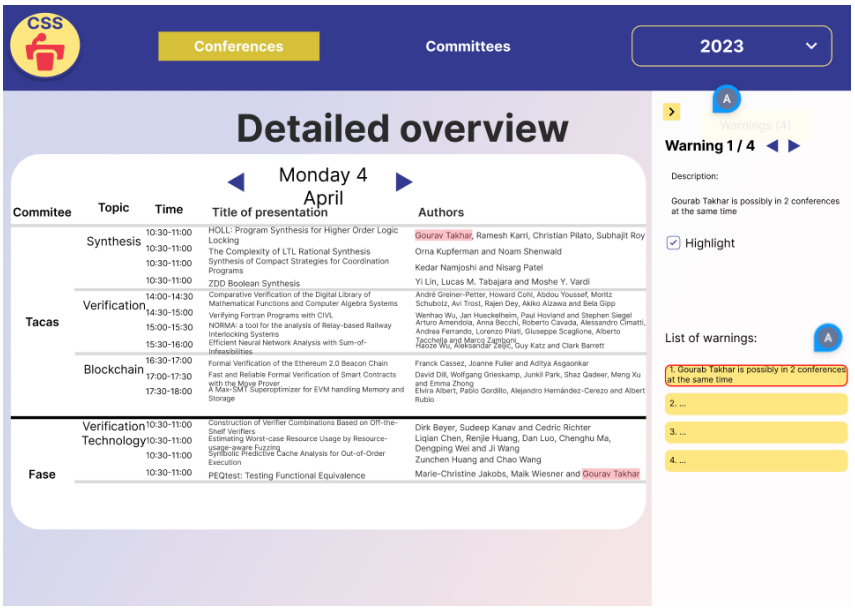


Figure 18: Detailed overview duplicate check

8.4.12 General Overview

A table overview, showing the conference schedule for the selected year in the navigation bar. This page could be reached from (Figure 19: Schedule creation) and would display the overview schedule for the specific year. Furthermore, the small table on the left part of the screen would indicate the allocation status for each committee (the number of assigned slots per timeslot) which would be dependent on the assigning slots page (Figure 6: Presentation slot assignment). Finally, the overview page (Figure 19: General overview schedule) was designed to have the option to export the schedule in a YAML format, so that it could be uploaded on the ETAPS website.

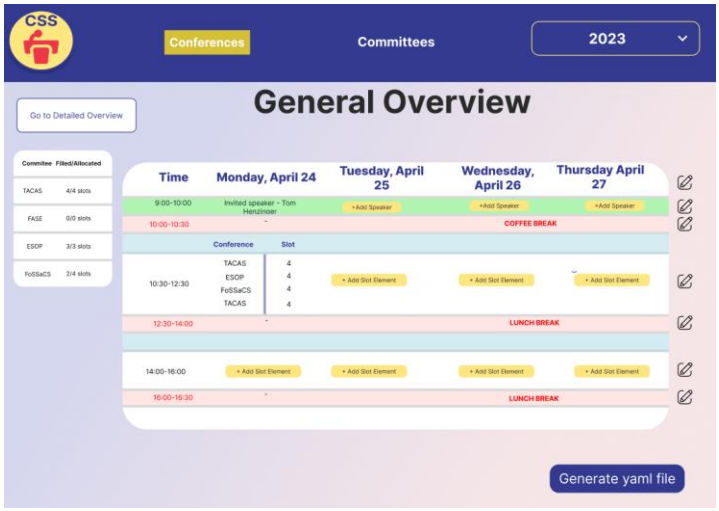


Figure 19: General overview schedule

8.4.13 Schedule Creation

When clicking on the “new” box for the conference creation from the main page (After successfully entering the credentials, the admin user would be able to choose either the Conference Scheduling System or the Committee Seat System (Figure 12: Main page). For each of them, an option would be provided to proceed with creating something new for either of the systems or loading existing data. Furthermore, in the navigation bar, the admin user would be able to easily switch between the 2 systems and change the year for which the actions are performed.), an empty table for the schedule would be loaded. The main possibilities for this page would include adding and editing slot elements, assigning speakers to slots, changing the row colours, depending on the activity.

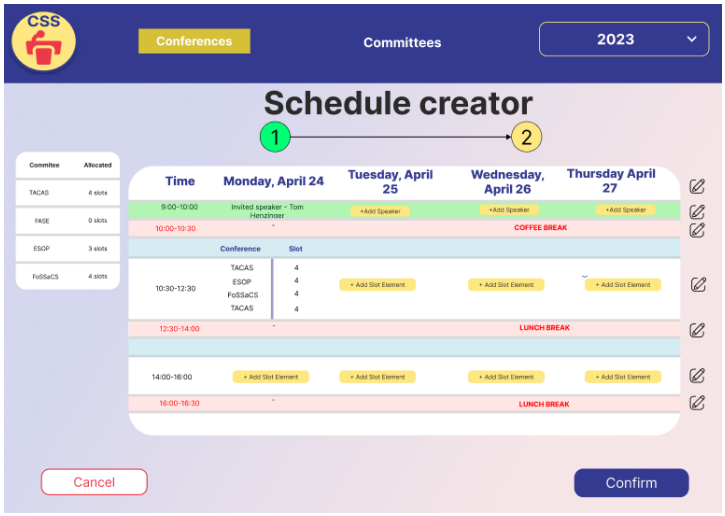


Figure 20: Schedule creation

8.4.14 Overview of the Committee members - per conference

A Page (Figure 21: Overview committee members), which would have the same functionalities as the committee chair page (Figure 8: Committee page), with the only additional feature that the admin user would be able to switch between the members per committee (TACAS, ESOP, etc.).

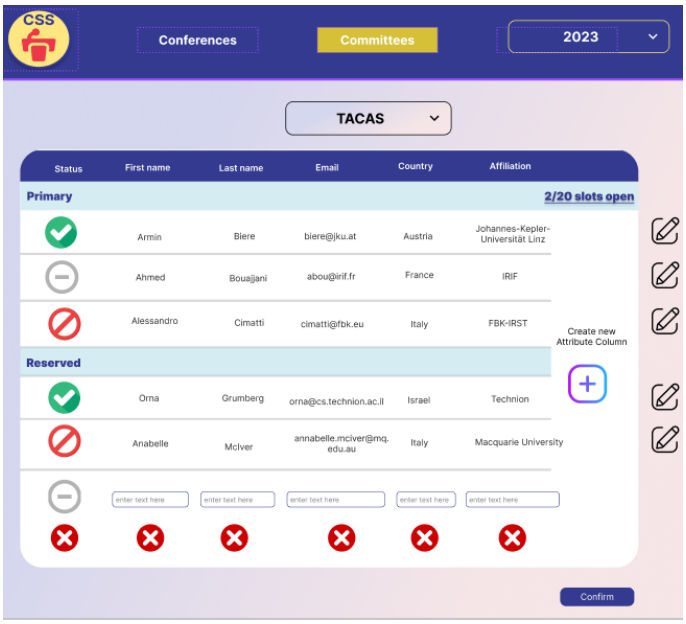


Figure 21: Overview committee members

9 Global System Architecture

In this section the overall global system architecture will be discussed and elaborated.

9.1 Overview

The System is a full-stack web application featuring a front- and back-end and a database as depicted in Figure 22: Overview of the System:

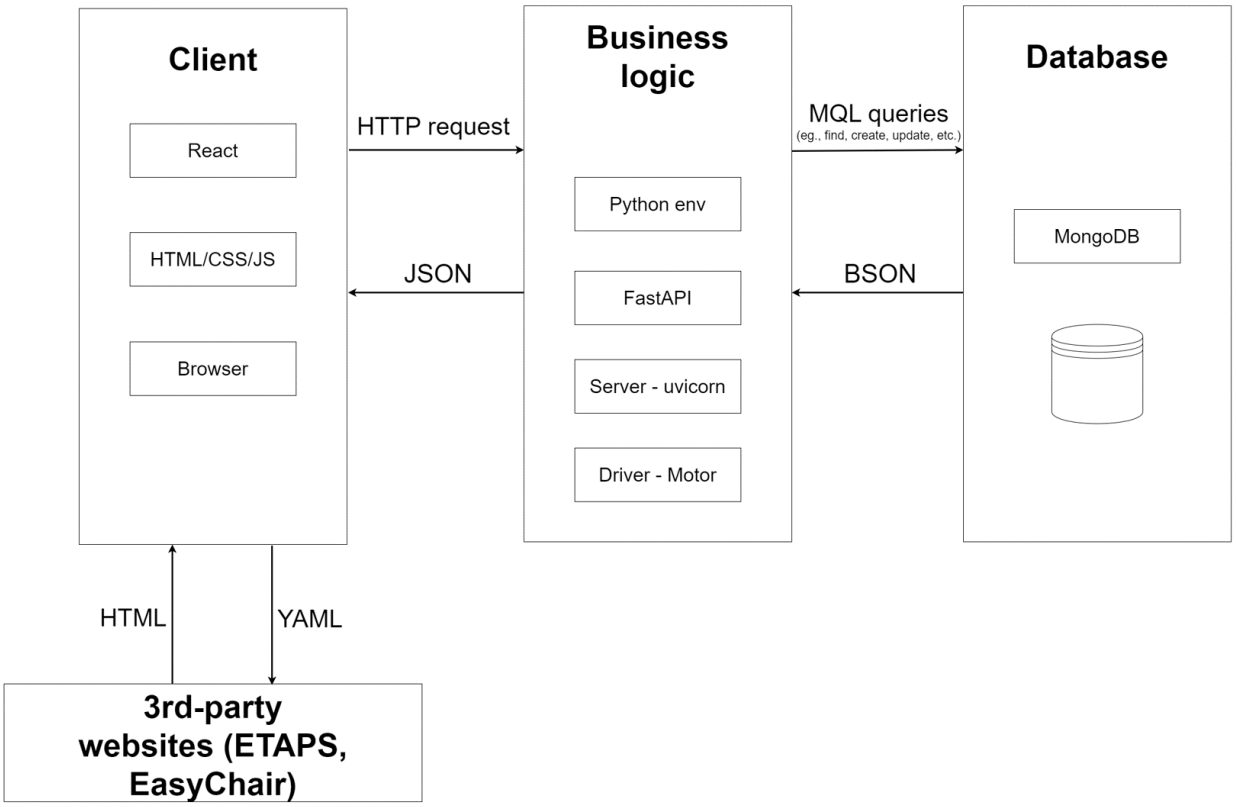


Figure 22: Overview of the System

As can be seen, our front- and back-end of the system will exchange utilizing HTTP requests containing JSON objects. The Database stores collections of BSON objects (Binary JSON) which will be serialized by the back-end ("business logic) in Python before directing it to the Client. Additionally, the client will export and import data from 3rd party websites: ETAPS and EasyChair to generate a table to display the schedule in YAML and it can receive data corresponding to accepted papers, which contain information about authors and papers and can be uploaded in the HTML format (which can be exported from EasyChair). A comprehensive overview of all the data in- and outputs that the system exchanges is provided in a table in Appendix E: Data Inputs and Outputs.

9.2 System Workflow

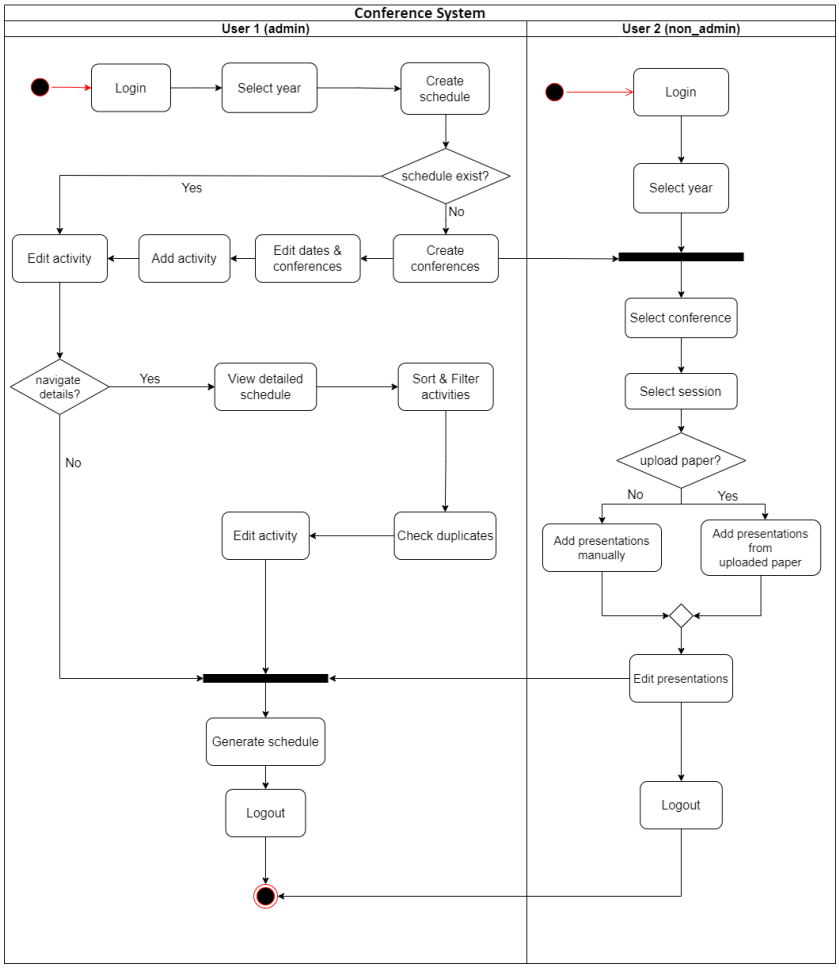


Figure 23: Conference system workflow

Figure 23: Conference system workflow depicts the workflow of the conference system. The system has two types of users: user 1, who is the steering committee chair and acts as the admin, and user 2, who is the committee chair. Before performing any operations, both users need to log in to the system. User 1 has the permission to create schedules and conferences in a selected year. Once conferences and sessions are created by user 1, user 2, who is a non-admin user, can select the corresponding sessions and add presentations either manually or by uploading HTML within his/her assigned conference. After adding presentations, user 2 can edit the start-end time, duration, title, and authors of existing presentations or delete any mistaken presentations. However, user 1 has additional privileges and can add other activities such as invited talks, plenaries, and breaks, etc. After editing the schedule, user 1 can decide whether to navigate the detailed schedule and edit possible duplicate authors in parallel time slots or generate the schedule in YAML format.

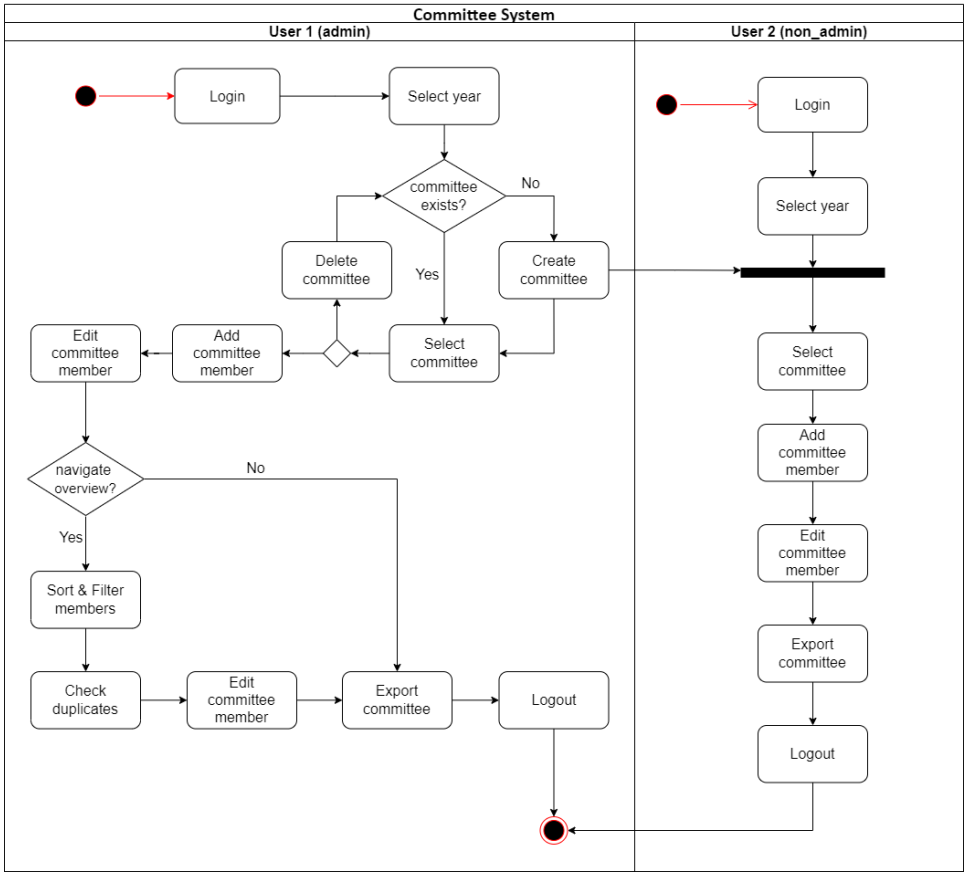


Figure 24: Committee system workflow

Figure 24: Committee system workflow depicts the workflow of the committee system, which also involves two types of users: user 1 and user 2. Similar to the conference system, both user 1 and user 2 need to login to get authenticated. After user 1 created committees, user 2 can add committee members, edit their invitation status, first/last name, email, country and affiliation, delete specific members within his/her corresponding committee, and export the accepted members in a consistent format used by EasyChair. For user 1, he/she can also delete unused committee as a table and navigate possible duplicate members across committees before export.

Additionally, user 1 who has the administrative privilege, can manage accounts of admin and non-admin users in both systems, such as creating new users, changing users' passwords and roles, and deleting users.

10 Front-end Architecture

10.1 Overall front-end architecture

One of the important aspects about the Conference Support System is that it unifies 2 systems, one for the Conference Scheduling, and another one for the Committee Selection process. Given the scope of the project and the fact that 2 systems are being developed, the development team had to spend a lot of time on identifying the requested capabilities of the systems and how to design them, so that they are scalable, maintainable, and improvable.

One of the important decisions that were made, included the flexible front-end architecture. By observing the proposed prototype and further identifying the needs of our client, the development team decided that many parts of both systems are shared and could be reused. Therefore, this was one of the reasons why a component-based framework such as React was chosen.

During development, some of the views from the mock-up were removed, new ones were added, and some were reordered. First, we removed the page from the admin view, in which the user can select between creating a new schedule from scratch or from template because we did not manage to implement the template creation of a schedule. Second, we decided to protect the non-admin pages with authentication too, because we are dealing with personal information, and we did not want to expose any data without protection. Third, in the mock up choosing a committee was the landing page for non-admin users but we decided first to have a page where the user can choose which of the two systems they want to use and then ask them for the committee. In addition, we realised we need to ask for the year in combination with the committee because from the user testing, we learned the users will most likely fill data for conferences in future years. The admin pages stayed close to the mock-up, we mainly added input boxes and buttons which were missed in the design phase. Highlighting the warnings in both systems was revisited and changed. We had problems with implementing highlighting but also, we decided that highlighting is not the best way to show the warnings when there are a lot of people/presentations in the table. That is the reason instead of implementing highlight we filter the table and leave only visible the entries that cause the warning. In hindsight, we believe that this solution is even better than the design with highlighting the entries in the table.

10.2 React

The front-end has been developed with React. In React components are the building blocks, and each page is a component which is made of smaller components. The App.js file is the root component of the application and that is where our routing is contained. Only the login page is publicly available and after logging in, depending on the type of account, the user is directed to the admin or non-admin pages.

Our files are split up in several folders:

- components – contains the components which are used in the pages.
- hooks – contains our custom hooks.
- context – contains context provider (used for implementing the security)

- API – contains the file where we initialise our HTTP client with which we communicate with the back-end.
- pages - contains the components for each page of the website.

This organization of the files, which was applied in most of the cases, offered the development team the opportunity to follow an approach like the OOP (Object Oriented Programming) paradigm, while using React components. This not only facilitated the whole development process, but also aided in producing a scalable, flexible, and extendable software which was one of the important implications for the Conference Support System from the beginning.

10.3 Roles

As mentioned in the current subsections, the Conference Support system has 2 types of users, an admin and non-admin. The former one is expected to be used from people who are responsible for the whole organization of the ETAPS conference, such as the SCC, whereas the later role is more dedicated towards CC.

10.3.1 Admin (Steering committee chair)

Given the input from the numerous meetings in the first 3 weeks with the client, the development team decided to use the following sitemap(Figure 25: Admin sitemap). It was designed in a way that facilitates the navigation from the one system to the other, which was a highly requested feature from the client.

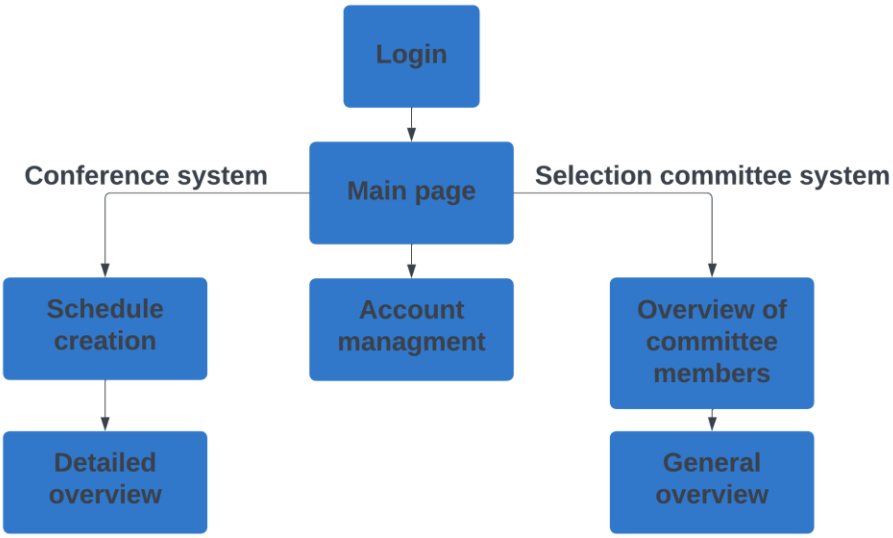


Figure 25: Admin sitemap

The website starts with a login page, where the admin should provide his credentials. After that there exist 3 possibilities.

One of them would be to access the conference system, to create or edit a schedule and receive a detailed overview with all the presentations for the year.

The second option would be to proceed with the selection committee system, which is possible to be done not only through the main page, but also trough the navigation bar, so that the selection committee process could be started.

Finally, it is important to have account management functionality for the people with admin role, therefore, from the main page it is possible to be proceeded to the dedicated page and edit, delete, or add new accounts. Initially, this was not a requested feature, however as the project proceeded, it had to be added for the ease of the client.

10.3.2 Non-admin (Committee chair)

For the non-admin role, the development team designed the sitemap (Figure 26: Non-admin sitemap) in a way which facilitates the work for one system, since it was agreed that non-admin users would not frequently change between the 2 systems. Therefore, starting from the main menu, the CC needs to choose one of the 2 systems.

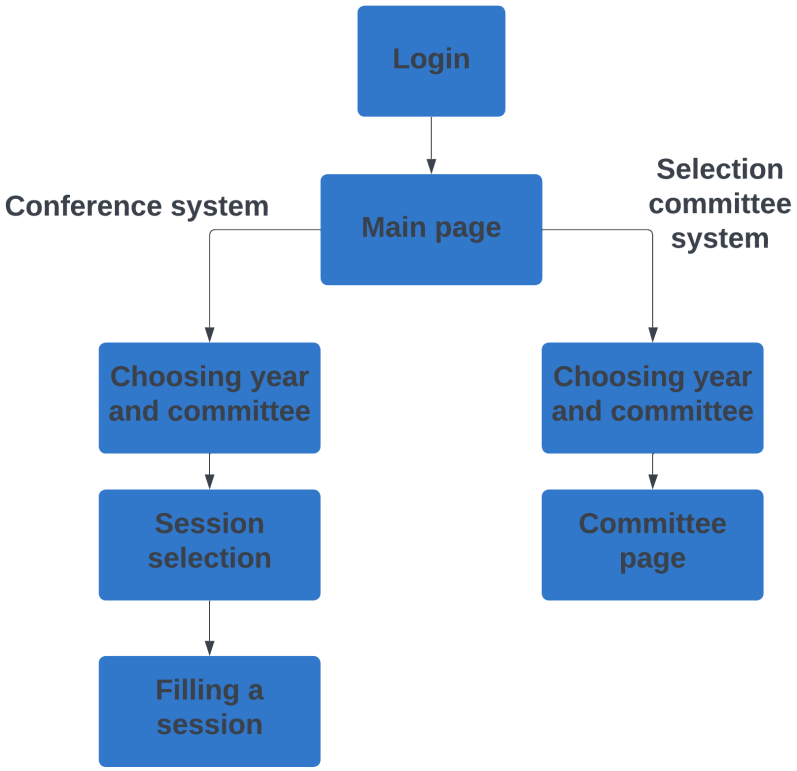


Figure 26: Non-admin sitemap

In case of the conference system, then the user could select session from the created ones by the admin, for which presentation speakers should be added.

For the selection committee system, it consists of one page, where the non-admin user adds participants' data per year and committee.

11 Back-end Architecture

The back-end is logically categorized into four main components: API, Data Models, Authentication, and Applications. Each category has a distinct area of responsibility and function, which will be elaborated upon in the following section of this report.

11.1 API

The API is the outward facing component of the system. Its main area of responsibility lies in combining all the other components into a single application and exposing the models as endpoints to the front-end.

Our API application uses the FastAPI web-framework and is written in Python. It is the entry-point and main module that hosts the other components and represents the back-end. The API app itself is logically separated into five modules:

1. Database – this module is responsible for establishing and maintaining the database connection. It is included on startup and called when querying the database.
2. Models – this module represents our data structure and models. The pydantic library is implemented to introduce an efficient means of typing and also verifying our model's validity (Since the Database itself does not enforce types).
3. Routers – this module manages the routing for the endpoints and provides the methods to perform CRUD operations on the models in the database. Here any function that is accessible to the front-end is exposed and all the HTTP requests will interact with the endpoints contained in the routers.
4. Export Files – this module is responsible for parsing the EasyChair exported HTML document that can be uploaded to the server (for extracting the names of authors and papers).
5. Tests – this module contains the tests that have been written and can be executed when the API was modified to ensure functionality.

Additionally, the API supports the OpenAPI specification standard [12] and comes with its own page that can be accessed by navigating to "<url: port>/docs" after successfully starting the main application. It utilizes swagger to display the endpoints, expected in- and outputs as well as the responses that the server generates. Finally, individual requests can be started from here, thus in case the API were to be extended this functionality can be made use of to test and integrate endpoints into the front-end.

A copy of the API documentation is provided in Appendix F: API Documentation, providing an overview of the individual endpoints, their parameters – including the security headers required to interface with them.

Technically, the API is a separate entity of the application as a whole – it could be used standalone and is only concerned with authentication and interfacing with the database.

11.2 Data Models

11.2.1 Modelling

The application contains models that represent the structure of the data of this system. The database does not enforce field types and does not perform validations, so it was essential for us to use a library which enables us to enforce data types as well as validate and parse data. An important factor here is that while the following section details our data schema, the API contains more models than are presented. For example, a “User” might have multiple variations of models, containing certain fields and having constraints. For example. a request to create a new user in the back-end must contain all the required fields necessary to populate the data, but a request to modify a user needs only to include the fields that are to be updated in the database. Hence, we would use two different models to validate the body of the request. A third model might be used to respond to a request – where “id” or “password” fields are not supposed to be included (i.e., for security reasons). This allows us to ensure the integrity of creating, reading, and updating the data. Additional features of the “Pydantic” library (used for our models) include validation and parsing of e-mail addresses. Due to the modelling, we are also able to use exceptions as responses, such as a “422 Validation Error”, which will return an insightful and useful message to the request, indicating the location and error that occurred if a request sent to the back-end is malformed. E.g., If a required field (such as a date) is missing, the API can respond with a validation error and the location of the expected field that is missing as part of the request [13].

11.2.2 Database

For our database, we use MongoDB – a document database. The database is comprised of loose collections (tables). Each collection (represented as a table in Figure 3) contains records which are key-value pair binary serialized object representations of our models (BSON). While BSON uses key-values pairs similar to JSON, it enables the use of explicit data-types such as datetime, which in JSON would have to be serialized as strings. The in- and outputs of the database are serialized to JSON and are instantiated as objects after receiving a response to a query in the back-end. By default, MongoDB does not enforce a specific schema for the tables, enabling the users to rapidly modify the structure and adapt models without having to migrate the database. This is the primary reason why we validate the models in the back-end when inserting or retrieving a document [14]. Additionally, the database is hosted on a cloud platform by default (and it is free) – enabling the team to work on a unified database without having to manually host it on every pc without effort. It could also be deployed offline (for free) – this decision however is left to the application owner.

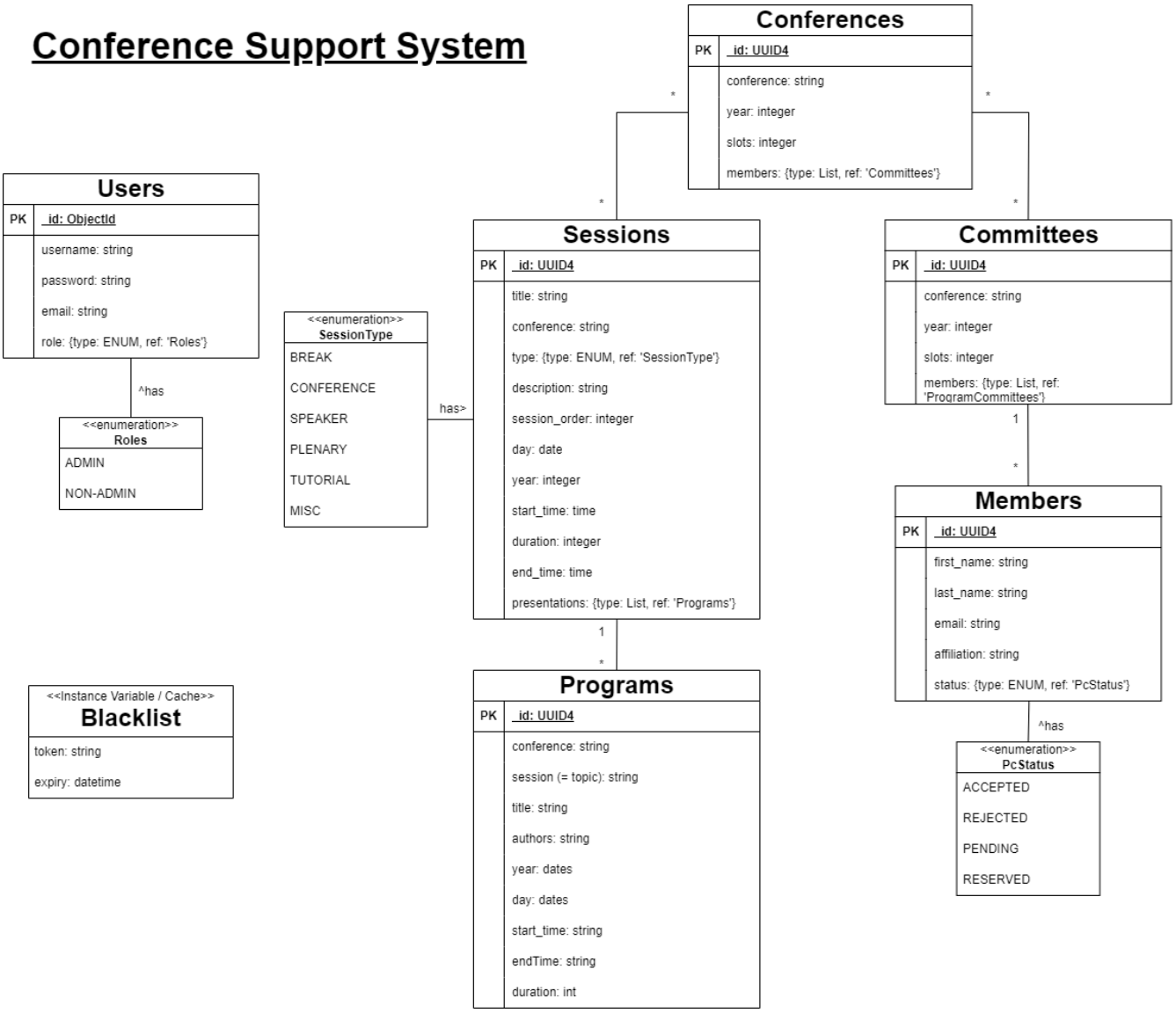


Figure 27: System Data Schema

11.2.3 Schema

The schema as depicted in Figure 3 represents the data-structure of the overall System. It not only includes the database component, but also a cache-based blacklist that is used for storing expired JWT's. The decision to use a cache-based system was made after deciding that querying the database for every request would be a performance hindrance and thus is locally instantiated.

The schema (as depicted in Figure 3) represents the data in a relational manner. By default, MongoDB does not enforce a schema so it was left up to us to experiment with how we store the data by use-case: Each session in the corresponding collection contains an array of UUID's (references) of the associated programs tables – a decision that was made due to the fact that the programs model itself will be adjusted and modified by the users of the system rather often,

hence we only have a relation in the form of object identifiers, which the back-end queries and combines before sending it to the front-end. The committees' models on the other hand contain a list of members objects. This decision was made because members are typically appended to a committee, in contrast to sessions. Both methods follow the best-practice standards of using MongoDB and we were free to make the choices without having a schema enforced upon us [15].

As can be seen in figure 3, each system has a many-to-many relationship to conferences – which is an intentional design choice made to enable the user to independently create multiple and unrelated conferences per year. So, for example, it might be the case that the information pertaining to the conferences differs from system to system, i.e., the SCC might be responsible for managing parts of the sessions for that year or parts of the committee management – since sometimes external conferences are combined into the yearly events, a flexible model was required. Every conference contains none to many sessions and each session includes none to many programs – a plenary session will not contain any programs, but rather have a title and description, whereas a conference session will not have a description but contain multiple programs.

11.2.4 Authentication

This module primarily concerns itself with authentication to request and modify data. A detailed description is provided in the next section "Security" of this report.

11.3 Applications

The final module represents the individual functions (applications) responsible for restructuring data into a new format, such as for the YAML and exports of sessions for the EasyChair website, parsing the names of members from the sessions and committee collections and parsing an HTML document (a list of accepted papers), which can be exported from the EasyChair website and conveniently uploaded to the system so that the CC's do not have to manually enter this information.

11.3.1 Duplicate checker for Sessions and Committees

The duplicate checker for sessions and committees uses a Python package "FuzzyWuzzy" which utilizes the Levensthein distance algorithm to calculate the similarity scores and determine the possible duplicates.

The duplicate checker for sessions required User 1 (the steering committee chair) to provide a year and a day as queries and compare the similarity of author names in parallel sessions, to determine whether the sessions are duplicate. For example, "J. Doe" and "John Doe" in sessions "10:15-10:45" and "10:30-11:00", respectively. Before checking the duplicate, session data will be pre-processed and sterilized. After that, the similarity scores of author names will be calculated by using FuzzyWuzzy, and these scores will be compared with a threshold defined by the user (85 by default) to determine the possible duplicate names. Because we only consider authors presenting in parallel time slots as duplicate sessions, we then will check whether the duplicate author names appear in overlapping sessions. If yes, the duplicate name ("John Doe") and the union of the overlapping time slots ("10:15-11:00" in our example) will be used as duplicate session indicators to inform User 1 on the webpage; otherwise, these sessions are unique.

The duplicate checker for committees requires User 1 (the steering committee chair) to provide a year as a query, and compare the similarity of the committee members' emails, first names and last names with a user defined threshold, which is 85 by default. If committee members have an identical email address, their first names and last names will be checked using the user-defined threshold. If they have similar first names and (or) last names, they will be considered as duplicates and a corresponding error message will be shown to User 1. If committee members do not have an identical email address but similar first names and last names, they will also be considered as duplicates and be reported.

11.3.2 YAML Export

The YAML export will generate a string of YAML objects that the front-end can then utilize to provide a file-download for the users. The main functionality herein lies to extract data from the existing structure into a new one (as provided by our client), so that it can be used to generate the schedule table on the ETAPS website. It was specified that this format is not finalized yet, so we decided to use the current structure (which we received) for primarily demonstration purposes.

11.3.3 EasyChair Exports

The export function concerning for the EasyChair website invitation system was a rather trivial implementation of extracting the list of members that have a "PENDING" status per committee and building a string (with a new line for each member) according to the specified format. The format changed multiple times throughout the development of the system and currently adheres to the most recently requested structure.

11.3.4 Parsing HTML

This function utilizes the "BeautifulSoup 4" library, which is primarily intended to be used for parsing HTML or XML documents (websites). Initially we experimented with various regular expression implementations but decided to use this library for simplicity's sake, to make this more easily extendible should the format of the HTML page change in the future, and because directly applying regular expressions was rather complicated in contrast to the current implementation.

12 Security

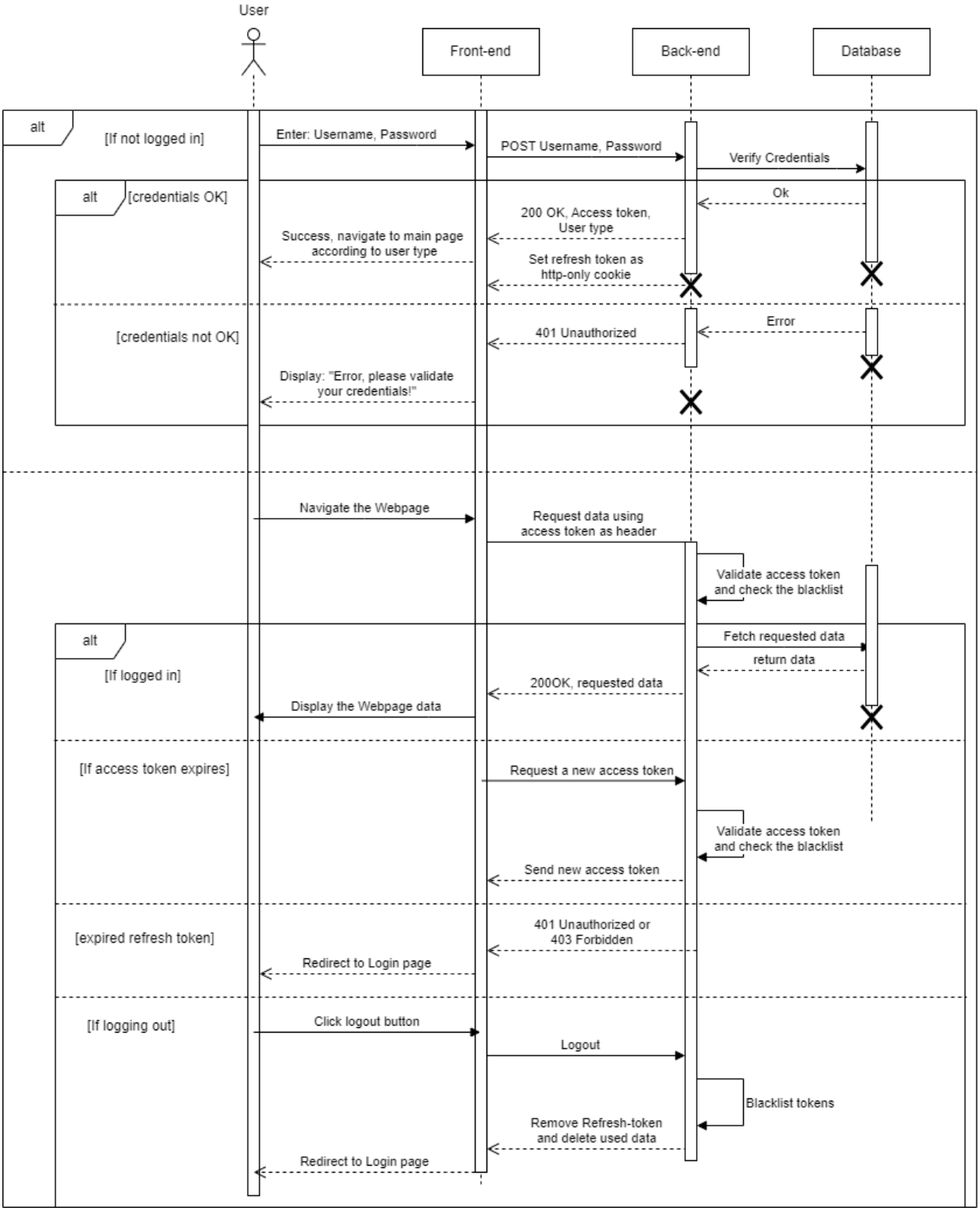


Figure 28: Authentication Sequence Diagram

The security aspect of the system is also very important since we must deal with not only the data of when a conference takes place and the scheduled presentations within a committee, but also store personal data of committee members such as First and Last name, personal email, affiliation, country of origin etc. Such data can be classified as sensitive, and a proper security system must be used to ensure that the availability of such data is secure.

Firstly, before using the system, the user is required to login into their account. The user credentials are then sent to the API to check the validity of the user. We also distinguish 2 different user types, an admin user, and a non-admin user. The main difference between the 2 types of users is that an admin user has a greater scope of influence on the system overall (such as creating a conference, editing the committees available or access to account management). The API ensures that the password corresponds to that of the user and the navigation system of the frontend ensures that the user is unable to navigate to pages to which it has no permissions.

Secondly, we decided to verify a user upon any data request using token-based authentication and have decided to follow the OAuth2.0 which is an open-standard authorization protocol. A token-based authentication protocol is safer since to validate the tokens, a secret key known only to the API is used. Thus, upon logging in, the user is provided with two different token types, an access token, and a refresh token. An access token has a short lifespan (1-2 hours) and is used solely for the purpose of requesting sensitive data from the API which verifies the token upon any requests that are intended to fetch, modify, or delete any data. Unlike the access token, the refresh token has a longer lifespan (up to 1 week), and the token is used to request new access tokens from the API. This is done to ensure that the user session is not lost upon the expiration of the access token, and while an access token is required to be set in the Auth header of the request, the refresh token is stored as an http-only cookie. The token is set as an http-only cookie to ensure that it is not accessible from the client-side, an http-only cookie is not accessible through client-side scripts and thus it helps mitigate hijacks of the refresh token (through XSS for example). Whenever the refresh token expires, the user is required to re-log into the system. The process of setting and refreshing access tokens is done in the background of the system and is invisible to the user.

The API also contains a locally stored blacklist. When a user logs out of the system, the refresh token is removed from the cookie storage and both the refresh and access tokens are stored in this local blacklist. This way, the user session is completely terminated and cannot be used anymore.

13 Testing

Testing is an important part of the development lifecycle. In order to guarantee a stable product, the development team dedicated time identifying the vulnerable parts of both systems and then to systematically test them through the development. Generally, the 5 methodological test types that were discussed within the team are: Manual, Unit, Integration, End-to-End and User testing. However, not all of them were feasible to be incorporated into the Conference Support System project. Therefore, in the following subsections, we will provide an explanation for all the test types that were considered and an argumentation why certain ones were omitted while for others more time was dedicated.

13.1 Test Plan

13.1.1 Manual

Manual testing is the most time-consuming type of testing, nevertheless it is suitable for both the front-end and the back end. For the former one, it includes predominantly clicking on available buttons, checking if the state of the webpage is changing as anticipated, trying to introduce some intentional bugs, and observing how the system will perform the error handling. Regarding the back-end manual testing, the main tools which were utilized were Postman together with Swagger. The former is a graphical user interface program dedicated for testing APIs, which includes building and sending various custom request types with parameters and then displaying the API response without the need of front-end. This allows for systematically testing individual requests while constructing the back end as well as commence initial performance tests i.e., how does the API handle multiple batch requests, how many concurrent requests and in what order are they completed etc.

Overall, both for the front-end and back-end, manual testing plays an important role throughout the whole development process. Not only does it provide an easy way of testing, but also a good impression on the way the system behaves. Furthermore, in our development team, manual testing was extensively used from the start until the end of the implementation span.

13.1.2 Unit

The unit testing concentrates on creating automated tests for single functionality both on the front-end and back-end. The main objective for unit testing was to check if a single part (unit) of the program performs as anticipated and identify edge-cases and fix possible mistakes related to them. For the back end, the in-built functionality of FastAPI utilizes the Starlette [16] and Pytest [17]. These libraries were mainly used during the back-end unit testing. For the Conference Support System, our team decided to avoid unit testing for the front-end part. In general, the main objective for front-end unit testing is to detect changes in the state of the webpage and compare the obtained results with the expected results. This is not only time-consuming, but also includes a variety of dependency issues which arise during the unit test writing process. Therefore, the amount of effort required to write unit tests for the front-end doesn't correspond to the quality of information that is received; hence unit tests were omitted for the front-end.

13.1.3 Integration

The idea behind integration tests is to ensure that various parts of the system can communicate and work together. Compared to unit tests which offer reliability and coverage but for a single functionality of the system, integration tests aim to test an integration of the individual units (functions) in a larger scope and in conjunction with each other, in order to determine if certain expectations pertaining to functionality, reliability, security and performance are met. Part of the important integration tests for the CSS project in the back end were to extensively verify the API routing and endpoints, as well as how it responds to: Correctly formulated requests, malformed requests, handling the data which could get sent as a POST request to the API and then fetched by a GET request after being modified by a PUT request. Additionally, the security mechanisms (JSON Web Token authentication) were also tested in conjunction with creating, reading, and updating data, as well as the response times of concurrent API interactions. Concerning the front-end, integration tests were also omitted for the reasons identified in the unit testing section. Overall, most of the focus is set on the visual aspects on the webpage (change of text, state of button, etc.) for connected parts, however the received value wouldn't outweigh the amount of effort spent on creating the tests.

13.1.4 End-to-End

Manual, unit, and integration testing offer good representation of the intended behaviour of the system. Unit and integration testing are a perfect solution for checking if individual parts of the system work as expected and whether they can communicate as expected. Nevertheless, those types of tests are not suitable to check if the whole system works as expected as a whole. For that reason, manual tests are performed, however they have one noticeable limitation, namely that they are time-consuming and tiring to perform. Therefore, End-to-end, also known as E2E, tests solve the problems. The main idea behind using them is to replicate certain user behaviour and to check if it is giving the expected results in different scenarios. There are many different applications for E2E tests to choose from and our team decided to use Selenium. It has many advantages such as multi browser support, ease of implementation, flexibility and more. Selenium enables us to test for individual requirements and if the application returns results according to the parameters. It is essentially an automated user performing a set of tasks consisting of our list of requirements to evaluate if they have been fulfilled and the application is functioning (and producing results) as intended. In conclusion, the E2E tests on the frontend offer the combined functionalities of both unit and integration tests, so focusing on E2E tests would bring significantly more value and allocate less time than the 2 testing types.

13.1.5 User testing

The end goal of each project is to deliver a product that meets the expectations of the client. This process is not straightforward, so to guarantee the success of the result, a proper usability testing approach was followed. The main idea behind this type of testing is to invite the client to use the current version of the product or to explain what functionalities were achieved in the span between the meetings, so that the client could share an opinion and confirm whether the development team is following the right direction. The advantages of user testing are numerous, including early spot of problems from design point of view such as the layout of pages and changing the states of buttons under certain conditions, to more functional ones, for instance adding an additional page

for the ease of the client, providing more functionality options for certain pages, or adding new feature which were not discussed during the requirements elicitation. On the contrary, the disadvantages include possible rewriting of existing code which could lead to delay and aiming to deliver new functionalities faster, in the span of 1 working week, so that it could be demonstrated to the client.

13.2 Test Results

13.2.1 Manual Testing

During the whole development of the Conference Support System project, there was a strong emphasize on the manual testing. For the front-end, most time was spent on checking visual updates (changing states, data, etc.), however it is difficult to summarize all the tests results in a comprehensible way. Therefore, it could be just stated that all the planned tests for front-end manual testing went successful and produced the expected results.

Moving to the back-end testing, 2 tools were used, namely Swagger and Postman. Both provide the functionality of checking whether the specified request receives a proper response. In the table below, the general data from manual back-end testing is provided.

Postman & Swagger manual testing		
Requests	**Level of importance**	**Test result**
- all POST request	H	Passed
- all DELETE request	H	Passed
- all PATCH requests	H	Passed
- all GET requests	H	Passed

Table 3: Manual Testing Results

13.2.2 Automated Testing

The tasks were split into 4 suits: Admin Conference creation system, Admin Selection committee system, Non-admin Conference system, Non admin selection committee system. Each suit has functionalities derived from the requirements. Each functionality has a level of importance which is ranked by High (H), Medium (M) or Low (L).

During the testing several bugs were found and fixed. One bug was that in the non admin page for the conference system the requests for getting the sessions did not include the selected year and therefore displaying the sessions incorrectly. Two other bugs were in the admin view, when editing a conference schedule or editing selection process, the backend was not correctly adding new committees to the database. And the last bug that was found was in the checking if there is a person in 2 presentations at the same time. The algorithm for that was not considering that a second presentation might start between the start and end time of the first presentation, because it was only comparing the start times and end times of the presentations and did not consider intersections.

We achieved 94% test coverage for the systems, with a total of 36 unit and integration tests executed. Our test covered all the major functionalities and edge cases, ensuring that the code meets our desired code quality standard and is robust enough to deploy.

For E2E tests, the shortcoming of testing with Selenium was that we could not test functionalities with uploading and downloading files. Therefore, testing the generation of the program in YAML format, uploading html of accepted papers, and adding presentation from uploaded papers was tested manually.

Admin Conference creation system		
Functionality tested	**Level of importance**	**Test result**
- create a conference with date range and committees	H	Passed
- add activity to a day and observe that allocated slots are updating correctly	H	Passed
- add many sessions to an activity	H	Passed
- edit conference dates/committees	M	Passed
- edit activity start time/duration/sessions	M	Passed
- delete activity	H	Passed
- have a detailed overview of all presentations for the whole program	H	Passed
- be able to filter for a day in the detailed overview	L	Passed
- have warnings for people that are possibly in more than one presentation at the same time	L	Passed
- generate YAML file with the program	H	Passed

Table 4: Test results - CC/Admin

As can be seen in Table 4, all important functionalities were tested and passed. This test section was mainly concerned with the OSCC creating a new conference and interacting with the data.

Admin Selection committee system		
Functionality tested	**Level of importance**	**Test result**
- start selection committee process with committees	H	Passed
- add a member for a committee	H	Passed
- edit a member	M	Passed
- change assigned slots for a committee	H	Passed
- edit list of committees for the selection process	H	Passed
- have a general overview of all members from all committees	H	Passed

- have warnings for people that are possibly in more than one committee	L	Passed
- if there are warnings be able to filter on those individuals	L	Passed
- export table in text file, formatted for EasyChair invite	H	Passed

Table 5: Test results - CS/Admin

As can be seen in Table 5, all important functionalities were tested and passed. This test section was mainly concerned with the OSCC creating a new conference committee and interacting with the data.

Non-admin Conference creation system		
Functionality tested	**Level of importance**	**Test result**
- select year and committee	H	Passed
- select a session	H	Passed
- change title of session	H	Passed
- add presentations to a session	H	Passed
- edit presentation duration/title/authors	M	Passed
- change order of presentation	H	Passed
- delete presentation	H	Passed
- add uploaded papers	L	Passed
- add presentation from uploaded papers	L	Passed

Table 6: Test results CC/Non-Admin

As can be seen in Table 6, all important functionalities were tested and passed. This test section was mainly concerned with the CCs interacting with a new conference schedule by creating, reading, updating, and deleting data.

Non-admin Selection committee system		
Functionality tested	**Level of importance**	**Test result**
- select year and committee	H	Passed
- add a member for a committee	H	Passed
- edit a member	H	Passed
- change assigned slots for a committee	H	Passed

Table 7: Test results - CS/Non-Admin

As can be seen in Table 7, all important functionalities were tested and passed. This test section was mainly concerned with the CCs interacting with a new conference committee list by creating, reading, updating, and deleting data.

13.2.3 User testing

On the 6th of March 2023, the development team performed a user testing together with the client. The main objectives were to identify whether the proposed prototype meets the needs of the client in terms of structure (the organization of the webpages) and if it is easy to use and intuitive. To test the objectives, the following 10 tasks were created:

1. Add a new speaker for the “Synthesis” session from the TACAS conference session on Monday 4th April and review the schedule afterwards.
2. Remove and add committee attribute(s) (First Name, Last Name, etc.)
3. Remove and add a selection committee member.
4. Create a conference schedule.
5. Change the time interval of the conference.
6. Edit a slot in the schedule.
7. Add a new slot element in the schedule.
8. Review the overall detailed schedule.
9. Start new committee selection process from scratch.
10. Review the overall proposed members and check the warnings.

While performing the actual user testing, the client had rather positive experience and managed to perform most of the tasks easily. Nevertheless, there was a rich and important feedback for each of the tasks. A summary of the proposed suggestions is shown in the list below:

- For the slot assignment page (Figure 6: Presentation slot assignment**)**, the “Synthesis” text should be editable.
- For the committee page **(**Figure 8: Committee page**)**, the columns for the table should be sortable.
- For the committee page **(**Figure 8: Committee page**)**, the status symbols and the corresponding column are not necessary.
- For the committee page **(**Figure 8: Committee page**)**, add a dedicated button for deleting a participant.
- For the committee page **(**Figure 8: Committee page**)**, add a button which moves person from reserve to active.
- For the general overview of the schedule page (Figure 19: General overview schedule), improve the page, so that the schedule is not the same for each day and more diverse information could be added.
- For the schedule creation pages (Figure 20: Schedule creation), different days might not have the same structure and organization.
- For adding/editing row element of the schedule, provide customizability with multiple possible options.

- For adding a slot element in the modal, provide an easier interface and possible explain better the idea behind the modal.
- For the new committee selection process (Figure 13: Creating a new committee.), possibly add the number of slots as an input.
- For the overview of committee members page **(**Figure 21: Overview committee members**)**, make sure other people could also access the page.
- For the general overview of committee members **(**Figure 14: General overview**)**, make a home page for people and provide and option to generate a HTML or CSV file.

14 Evaluation

In this section our team will reflect on the overall outcome of the design project. The successful parts will be discussed, alongside the limitations of the current solution. Finally, the possible future improvements are going to be suggested.

14.1 Success

From the first day of the design module, all team members had the goal of delivering a scalable, maintainable, and extendable solution. In the end, it could be stated that the objective was achieved. Furthermore, it was the result of fulfilling several important aspects.

14.1.1 Teamwork

It is almost sure that not the entire process of developing a software would proceed smoothly. One way of mitigating this risk is teamwork. Luckily, all the team members were dedicated, motivated and approachable people, so throughout the whole design project, all team members had high spirit and were eager to help each other. Teamwork could be identified as one of the key points of the development team, which ensured in delivering a satisfying product.

14.1.2 Communication

The regular communication not only with the client, but within the team is important for each project. Thus, the development team had daily meeting sessions and weekly meeting sessions with the client which helped in understanding better the project. Therefore, it can be clearly stated that the communication was one of the best-performing aspects of the development team throughout the module.

14.1.3 Organization

In many cases projects become overwhelming. Many different requirements are requested from the client, some of them are later changed, so organizing a software project might be a challenging task. Gladfully, all the team members were prepared for the upcoming chaos and managed to organize everything accordingly. With the help of Discord, SharePoint, and Trello, all the important tasks and information were placed into the correct resource, which facilitated the process. Overall, the organization was on a high level from the beginning of the module until the end.

14.2 Limitations

As every software product, the Conference Support System has certain limitations to it. The development team managed to identify 2 most general limitations, namely the complexity and scope of the current solution.

14.2.1 Complexity

The complexity of the project was rather high. This came from the fact that the requested solution for conference support system was a rather unique project, therefore, all the team members needed to research and investigate for appropriate solutions. The team needed several weeks in the beginning to fully grasp the client's requirements for the system and the way the users of the system would interact with it. Eventually, the team and the client got to the same level of understanding but the consequence of that is that the design phase took longer than predicted.

Additional to the aforementioned points, we've been constantly updating and re-evaluating requirements. Because of the inexperience with the new frameworks which we intentionally chose (aiming to gain additional experience) it took more time for us to implement the basic application structure, with the aim of having a modular and extendable product suitable for usage and extension in the next years. Overall, we are happy to report that we are satisfied with the end result, but this was certainly a circumstance that increased the difficulty significantly.

14.2.2 Scope

Given the fact that the development team worked on 2 systems, certain limitations are connected to the scope of the project. It was unfeasible to spread the project around multiple requirements and functionalities, since the time was limited (10 weeks) and there was a technical dept within the team members (most of the used technologies had to be self-learnt from each developer). Thus, the scope of the project was restricted, but covered over 90% of the identified requirements. The team focused on implementing all the functionalities and therefore there is room for improvement on the UI and the performance aspects. We believe that we have laid a solid groundwork for this application and perhaps made it a bigger system than originally envisioned by the client, but we demanded this from ourselves, and it was also our choice, because we are passionate and want to be satisfied with the work we delivered – this was our goal from the start.

15 Future Works

The current version of the Conference Support System should serve the basic needs of the client. Nevertheless, some future improvements could be made to the website which could divided into 2 parts, development, and deployment.

15.1.1 Development

In terms of development, the current solution could be improved in many ways. Starting with the performance, it could be easily improved, for example, by speeding certain API calls and removing unnecessary complications in the program.

Another improvement could be the user interface. Currently it is simplistic and not eye-catching, therefore with the correct designing skills and tools, a future developer could turn the existing

website from an ordinary-appearing Conference Support system to a completely new, visually pleasing website. This would be a difficult task and should be tackled by people with designing touch and experience, otherwise the improved version might be worse compared to what is currently delivered.

Finally, as mentioned earlier, the system is complex. There are many options for extending it and implementing new functionalities which would be further identified after the client utilizes the current solution. For instance, generating the committee members in a YAML format is one functionality which might be appropriate to be implemented. Furthermore, the detailed pages with presentations for the Conference Scheduling system could be improved and extended, so that both pages show the data in a more comprehensible way and more editing options are provided.

15.1.2 Deployment

Since the client has an interest in deploying the application, we have included some considerations that must (in our opinion) be considered before deploying this project in a production environment.

15.1.2.1 Front-End

1. The front-end react application should be compiled before deployment. For detailed and up-to-date instructions for deploying the app, please refer to the official documentation [18].
2. Another recommendation for deployment is removing the REACT developer tools in the browser. This way, the users of the website cannot access any information about the REACT components used in the project, this is a big enhancement of security. This can be accomplished with the usage of a publicly available library. For more details on how to turn off REACT developer tools, refer to [24].

15.1.2.2 Back-End

1. Environment Variables: Currently the connection string and password to the database is stored in plain-text. This is not ideal but was a requirement to transfer the code to the client easily (without having the hassle to export or re-setup the environment, making it more complicated). Before deployment, sensitive data, including the app configuration settings should be replaced with environment variables.
2. Versioning: The Backend has been developed and tested using FastAPI version 0.92.0 on Python version 3.10 (LTS). Due to changes in the annotation system, we have already discovered that there will be issues (due to syntax) caused in newer versions of Python and FastAPI.
3. The framework and dependencies versions are in the requirements.txt file of the backend directory. To guarantee proper functioning of the system, the versions must be adhered to.
4. The backend has been tested running uvicorn server, a high performance ASGI web server implementation for Python [19]. This solution can also be utilized for actual deployment scenarios, but it is not limited to it. Some noteworthy alternatives are

Hypercorn [20] and Daphne [21] - a deployment using an Apache Web-Server is also possible but not officially supported.
5. The final version of the product does not come with HTTPS enabled or tested. Due to a lack of resources this was not possible for our team and must considered to be implemented if the domain is to be exposed to the public. Additionally, to generating HTTPS certificate and providing it to the FastAPI app parameters, an automatic redirect for all HTTP requests to HTTPS should be implemented.
6. In the main.py file, CORS origins must be adapted to the location of the React front-end server, as it is currently expecting the front-end to communicate from http://localhost:3000.
7. A manual for deploying FastAPI can be accessed on the official website [22]

15.1.2.3 Database

1. The final version of the product utilizes a cloud instance of a MongoDB database. The URL and access parameters will be transferred, but before deployment it must be considered where the data is stored (also regarding legal obligation),
2. By default, MongoDB is compliant with GDPR standards, which makes it feasible to continue using the database (as-is) or replacing it with another free-tier instance. It is also possible, to deploy a MongoDB database locally on Mac, Linux, or Windows systems [23].
3. Before deployment, the database connection string located in the “backend/app/db/settings.py” file should be transferred to a reference to an environment variable.

Apart from the abovementioned points, it is possible (but not advisable) to just simply deploy the system *as-is*. The product is functional, and the abovementioned considerations are out of scope of this design project.

16 Conclusion

Everything has an end. In this final section of the report, the development team would mainly focus on providing details about the final status of the requirements in the end product, why some of them were changed or removed.

16.1 Requirements

After the final requirements elicitation process in the first weeks of the module, a total of 40 requirements were identified, 22 of them being “must”, 12 being “should” and 6 being ”could”.

16.1.1 Iterative process

Throughout the module, the development team followed an iterative approach towards the weekly meetings with the client. This resulted in changing or removing some of the initially set requirements, due to complexity problems, or, due to the client switching the priorities of certain requirements. Therefore, the team had to be flexible and manage to satisfy the needs of the client, so that she could receive a product which maximally meets her expectations.

16.1.2 Changed Requirements

Since the wording of some requirements implied a concreate way of implementing them, it was discussed with the client that some of the requirements could offer the same functionality, however with a different implementation. A notable example are all the requirements related to duplicate checking and highlighting (RE8, RE9, R15, RE24 from Appendix B: Requirements) were not implemented with highlighting functionality, but with a filtering option achieved by pressing a button on the sidebar of the dedicated pages.

Regarding the inconsistent input requirement (RE16 from Appendix B: Requirements), it was a accomplished by providing a form with dedicated fields for name and surname, so that the format could be standardized.

16.1.3 Removed Requirements

Initially, the development team aimed in facilitating the schedule creation process of our client, thus a requirement to create a schedule from template was identified as “must” (RE6 and RE29 from Appendix B: Requirements). After a discussion with the client in the final weeks of the module, it was agreed that the aforementioned requirement could be removed.

Another set of requirements that aimed in automating the work of the client were generating the list of committee members in YAML format (RE19 from Appendix B: Requirements) and automatically replacing a person who changes from “pending” status to “declined” with another one from the reserve list (RE22 from Appendix B: Requirements). For the latter one, currently there is an alternative provided in the final product which does that with one of the people from the reserve list, but not necessarily from the top one.

16.1.4 Completed Requirements

In the end, the development team managed to complete all of the abovementioned requirements, excluding the ones which were removed.

16.2 Acknowledgements

In the end of this report, our team would like to share few words regarding the whole process and project. We would like to thank our client and supervisor Prof. Dr. Marieke Huisman for providing us with the opportunity to do the Conference Support system project. We have learned a lot of important lessons by doing it, not only from development perspective, but also from an organizational one. The project was challenging and with various possibilities for implementation, however in the end, we hope that a suitable and extendable prototype solution is provided that should facilitate the work of Prof. Dr. Huisman. Then we would like to thank Jan Kofron for the discussion we had about generating the conference program in YAML format and deploying the system. Thank you for the efforts from all parties, hopefully in the end we managed to help you by producing a useful system.

17 Bibliography

[1] ETAPS about. (n.d.). ETAPS. Retrieved April 21, 2023, from https://etaps.org/about/etaps-conferences/

[2] Fairley, R.E. & Willshire, M.J.. (2003). Why the Vasa Sank: 10 Problems and Some Antidotes for Software Projects. Software, IEEE. 20. 18- 25. 10.1109/MS.2003.1184161.

[3] Horizontal vs. vertical cloud scaling: Key differences and similarities. (2022, August 5). Spiceworks. https://www.spiceworks.com/tech/cloud/articles/horizontal-vs-vertical-cloud-scaling/

[4] Introducing FARM stack - FastAPI, React, and MongoDB. (n.d.). MongoDB: The Developer Data Platform | MongoDB. https://www.mongodb.com/developer/languages/python/farm-stack-fastapi-react-mongodb/

[5] Figma. (n.d.). https://www.figma.com/.

[6] Kopf, B. (n.d.). The Power of Figma as a Design Tool. Toptal. Retrieved April 21, 2023, from https://www.toptal.com/designers/ui/figma-design-tool

[7] What is version control? (n.d.). GitLab. Retrieved April 21, 2023, from https://about.gitlab.com/topics/version-control/

[8] Stack Overflow. (n.d.). Version Control Leaderboard. Retrieved April 21, 2023, from https://survey.stackoverflow.co/2022/#technology-version-control

[9] What is kanban? (n.d.). Atlassian. Retrieved April 21, 2023, from https://www.atlassian.com/agile/kanban

[10] The Upwork Team. (2020, October 23). 6 Most Common Software Development Methodologies. Upwork. Retrieved April 21, 2023, from https://www.upwork.com/resources/most-common-software-development-methodologies

[11] Lucid Content Team. (n.d.). What the Waterfall Project Management Methodology Can (and Can't) Do for You. Lucidchart. Retrieved April 21, 2023, from https://www.lucidchart.com/blog/waterfall-project-management-methodology

[12] OpenAPI Specification v3.1.0. (2021, February 15). OpenAPI Initiative. Retrieved April 21, 2023, from https://spec.openapis.org/oas/v3.1.0

[13] Handling Errors. (n.d.). FastAPI. Retrieved April 21, 2023, from https://fastapi.tiangolo.com/nl/tutorial/handling-errors/

[14] Databases and Collections. (n.d.). MongoDB. Retrieved April 21, 2023, from https://www.mongodb.com/docs/manual/core/databases-and-collections/

[15] Model Relationships Between Documents. (n.d.). MongoDB. Retrieved April 21, 2023, from https://www.mongodb.com/docs/manual/applications/data-models-relationships/

[16] Introduction. (n.d.). Starlette. Retrieved April 21, 2023, from https://www.starlette.io/

[17] Pytest: helps you write better programs. (n.d.). Pytest. Retrieved April 21, 2023, from https://docs.pytest.org/en/7.2.x/

[18] Deployment. (n.d.). Create React App. Retrieved April 21, 2023, from https://create-react-app.dev/docs/deployment/

[19] Introduction. (n.d.-b). Uvicorn. Retrieved April 21, 2023, from https://www.uvicorn.org/

[20] Contents. (n.d.). Hypercorn. Retrieved April 21, 2023, from https://pgjones.gitlab.io/hypercorn/

[21] Django. (n.d.). Daphine. GitHub. Retrieved April 21, 2023, from https://github.com/django/daphne

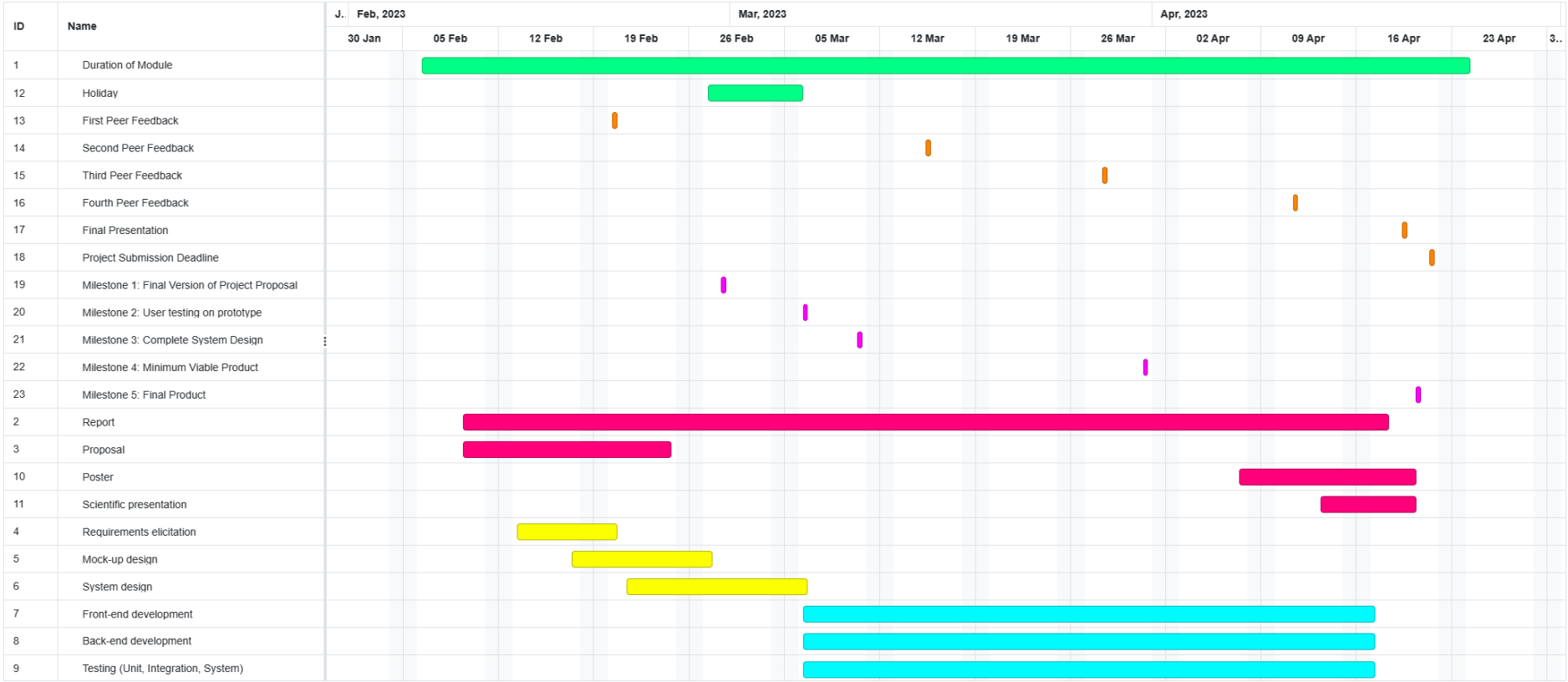
[22] Deployment - Intro. (n.d.). FastAPI. Retrieved April 21, 2023, from https://fastapi.tiangolo.com/deployment/

[23] MongoDB Community Kubernetes Operator Download. (n.d.). MongoDB. Retrieved April 21, 2023, from https://www.mongodb.com/try/download/community-kubernetes-operator

[24] disable-react-devtools. (n.d.). Npm. Retrieved April 21, 2023, from https://www.npmjs.com/package/@fvilers/disable-react-devtools

18 Appendix

18.1 Appendix A: Planning overview



18.2 Appendix B: Requirements

Modal Verb	**Definition**
Must	Mandatory for the project
Should	Adds significant value but is not vital
Could	Small impact, Nice-to-have
Actor	**Definition**
User 1	Overall Steering Committee chair of the ETAPS conferences
User(s) 2	Individual ETAPS conference chair members
Abbreviation	**Name**
System 1	Conference Scheduling System
System 2	Committee Seat System

System 1		
Nr.	**Description**	**Type**
RE1	System 1 *must* let User 1 create a multi-conference schedule template for all conferences.	Functional
RE2	System 1 *must* support timeslot allocation ranging from 15 to 120 minutes	Functional
RE3	System 1 *should* support three types of timeslot types: Presentations, misc. (E.g., Breaks or Tutorials), Plenary Sessions (E.g., Breaks, Plenary Sessions)	Functional
RE4	System 1 *should* support the generation of plenary sessions that are not assigned to a concrete structure.	Functional
RE5	System 1 *must* support multiple sessions occurring for individual conferences simultaneously	Functional
RE6	System 1 *must* let Users 2 fill in the created template for their respective conference.	Functional
RE7	System 1 *must* let User 1 review the multi-conference schedule once all Users 2 have provided data.	Functional
RE8	System 1 *must* output a general and normalized overview of the schedule once all Users 2 have provided data.	Functional
RE9	System 1 *must* output an overview of the programs (multi-conference schedule), indicating possible duplicate entries, or typing mismatches (e.g., the same name typed in two different formats or possibly mistyped) for each conference schedule.	Functional
RE10	System 1 *should* generate the schedule in the YAML format, which is usable for the ETAPS website to display the conference tables.	Functional
RE11	System 1 *must* provide an overview of the number of assigned slots per conference.	Functional

RE12	System 1 *could* allow Users 2 to upload their Easychair html file containing a list of accepted papers and speakers,	Functional
RE13	System 1 *must* allow User 1 to create multiple parallel sessions for conferences individually.	Functional
System 2		
Nr.	**Description**	**Type**
RE14	System 2 *must* provide a form for all Users 2 to enter data for the committee for each conference.	Functional
RE15	System 2 *must* show an over overview/highlight of overlaps in the names (all occurrences of the same name and surname must be overviewed/highlighted, e.g., multiple occurrences of Alex Petrov should be indicated) between the committees of the four conferences.	Functional
RE16	System 2 *must* provide an overview/highlight of inconsistent inputs (possibly following different format than what the system anticipated, e.g., ***Name Surname*** instead of ***Surname Name*** or having a close spelling e.g., Alex Petrov and Aleks Petrov) in the names between the committees of the four conferences.	Functional
RE17	System 2 *must* generate the information in a specified format: FirstName LastName <E-Mail> which is required as input to the Easychair website to manage committee members invitations.	Functional
RE18	System 2 *must* take input whether the participants accepted or rejected the invitation.	Functional
RE19	System 2 *should* generate the committees in the YAML format, which is usable for the ETAPS website to display the committees.	Functional
RE20	System 2 *must* allow Users 1 and 2 to add new names	Functional
RE21	System 2 *must* allow Users 2 to enter a reserve list of potential program committee members	Functional
RE22	System 2 *must* indicate members assigned to multiple committees	Functional

Combined Systems		
Nr.	**Description**	**Type**
RE41	The systems should let Users 1 and 2 edit data whenever it is displayed	Functional
RE23	The systems *must* let Users 1 and 2 correct possible typing mistakes after entering the data	Functional
RE24	The systems *could* provide an additional summary overview of highlighted issues for the committee seats and conference programs which User 1 can iterate through and verify.	Functional
RE25	The system *should* support at least two types of users: User 1 and User 2.	Non-Functional
RE26	The system *must* have a login/authentication system such that only permitted users can access it.	Functional
RE27	The system *must* store personal data securely.	Non-Functional
RE28	The systems *must* have a front-end (interface).	Functional
RE29	The systems *should* provide/allow the creation of templates with smart default selections that can be adjusted by User 1.	Functional
RE30	The systems *should* allow the users to enter information in as few (but dedicated) fields as possible to prevent mistakes.	Functional
RE31	The systems *could* store the information in a database.	Functional
RE32	The systems front-end *should* support Chromium based.	Non-Functional
RE33	The systems' front-ends *must* support the Firefox browser.	Non-Functional
RE34	The systems' front-ends *should* support the Safari browsers.	Non-Functional
RE35	The systems' front-ends *must* work on a laptop or desktop device.	Non-Functional
RE36	The systems front-end *could* work on a Smartphone.	Non-Functional
RE37	The front-end *could* support colour-blind correction.	Non-Functional
RE38	95% of the users *should* understand the purpose of the specific part of the web-app in less than 30 seconds.	Non-Functional
RE39	The systems *should* be available 99% of the time.	Non-Functional
RE40	The systems *could* support data-upload via spreadsheets	Functional

18.3 Appendix C: User Stories

Requirement (Nr.)	**User-Story**
System 1	
RE11	As a Steering Committee chair, I want the system to be able to provide a visual interface which has information regarding the number of available slots per conference.

RE4	As a Steering Committee chair, I want to be able to create a schedule and assign slot of time for multiple parallel conferences
RE3	As an ETAPS Conference speaker/presenter, I want the system to be able to give one presentation to one timeslot at a time across all conferences.
RE3	As an Individual ETAPS conference chair member, I want to be able to assign speakers to sessions for my conference.
RE7	As a Steering Committee chair, I want to have a clear overview of the program after all the chairs have added their conference information.
RE7	As a Steering Committee chair, I want to be able to change the information about the sessions after all the chairs have added their conference information.
RE9	As a Steering Committee chair, I want the system to overview/highlight possible duplicate entries.
RE8	As a Steering Committee chair, I want to be able to clearly see (with the help of highlighting) when people are present in multiple conferences simultaneously (duplicates).
RE10	As a Steering Committee chair, I want to generate the conference program in the YAML format, which is usable for the ETAPS website to display the program.
RE2	As an Individual ETAPS conference chair member, I want to be able to specify that a presentation can last from 15 to 120 minutes.
System 2	
RE14	As a chair member of one of the ETAPS conferences, I want to be able to enter the data in a form on the website.
RE15	As a Steering Committee chair, I want to identify whether steering committee members are assigned to multiple committees simultaneously.
RE16	As a Steering Committee chair, I want the system to output an overview/highlight of possible inconsistent inputs in the names between the committees four conferences.
RE17	As the Easychair organization, I want to receive the list of people in a specific format (*FirstName LastName* <*EmailAddress*>), so that I can send the invitations automatically.
RE19	As a Steering Committee chair, I want to generate the committees in the YAML format, which is usable for the ETAPS website to display the committees.
RE18	As the steering committee chair, I want to be able to indicate whether potential steering committee members have accepted or rejected the invite from the Easychair website.
RE18	As an individual ETAPS conference chair member, I want to be able to indicate whether potential steering committee members have accepted or rejected the invitation from the Easychair website.
Combined Systems	

RE28	As a Steering Committee chair, I want to be able to interact with the data visually.
RE28	As an individual conference chair member, I want to be able to interact with the data visually.
RE26	As a Steering Committee chair of the ETAPS conferences, I want only people with permission to have access to the data on the website.
RE24	As a Steering Committee chair of the ETAPS conferences, I want the system to be able to impose people on entering information correctly.
RE29	As a Steering Committee chair, I want to conveniently enter the data in templates with smart default selections.
RE23	As a Steering Committee chair, I want to modify data templates if they need to be adapted.
RE29	As an individual ETAPS conference chair member, I want to conveniently enter the data in templates with smart default selections.
RE30	As a Steering Committee chair of the ETAPS conferences, I want to conveniently enter the data in as few fields as possible
RE33	As a Steering Committee Chair, I want to be able to access the website with my Firefox Browser.
RE33	As an individual ETAPS conference chair member, I want to be able to access the website with my Firefox browser.
RE34	As a Steering Committee Chair, I want to be able to access the website with my Safari browser.
RE34	As an individual ETAPS conference chair member, I want to be able to access the website with my Safari browser.
RE35	As a Steering Committee Chair, I want to be able to access the website with my laptop or desktop computer.
RE35	As an individual ETAPS conference chair member I want to be able to access the website with my laptop or desktop computer.
RE25	As a Steering Committee chair, I want to be able to use both systems.
RE25	As an individual ETAPS conference chair member, I want to be able to use both systems.
RE27	As a user of the systems, I want my personal data to be stored encrypted.
RE32	As a user of the systems, I expect both to support Chromium based browsers.
RE39	As a user of the systems, I expect them to be available 99% of the time.

18.4 Appendix D: Meeting Reports (in chronological order)

Meeting Report (Week 1 - 09.02.23)

In the first meeting the group and client introduced each other and established an initial framework on how to structure the project organization. We agreed to have weekly meetings (Tuesday evening, 17:00h at Clients office).

The client is the overall Steering Committee chair of the ETAPS conferences and specified two distinct issues related to her responsibility, both requiring a solution to facilitate and optimize their workflow. Additionally, each individual conference has 1 or 2 chairs – these are the people that the client interacts with primarily.

The first issue regards the creation of a conference program and introduced us to the process they are currently using:

1) The client receives a program for each individual conference containing a program outline consisting of information about the conference participants (the speakers):
    a. Name,
    b. Affiliation,
    c. Title of presentation.
    d. Length of presentation
2) Additionally, the program outline consists of sessions assigned to a specific time slot. Each session consists of several presentations. The client proceeds to fill in the information and generate a program according to a specific format.
3) The client then verifies the data quality of (1) and corrects it manually if needed.
4) Utilizes the spreadsheet to generate a proposed program for the four conferences, by calculating the timeslots,
5) Manually verifies the program to check: Overlap of speeches and timeslots so that they do not conflict each other in the four conferences,
6) If all is okay, shares the program and uploads it to the webpage in (conjunction with the administrator of the webpage).

The second issue regards the proposal of a selection committee for each conference, here the client currently uses the following process:

1) The client receives a list with proposed candidates and information about:
    a. Name,
    b. Affiliation,
    c. E-Mail,
    d. Personal website,
    e. Information i.e., Gender, Demographics, Industrial background et cetera.
2) Generates a selection committee (format),
3) Verifies the information, i.e., checking for duplicates and overlap between the potential members (such that they are not in more than one committee simultaneously),
4) Once verified by the committees, invites the candidates by using the website Easychair and provided contact information,
5) Verifies which members accept the invitations,
6) Once finalized, displays the information pertaining to the committee members on the ETAPs website (in conjunction with the administrator of the webpage).

The client emphasizes that in both issues, a lot of tedious and manual work is involved. First, when the members sign up (provide their information), they do not necessarily adhere to the formats of the fields that they are supposed to fill in – potential issues can occur, i.e., information might be missing, duplicated or the fields themselves might be modified. Second, the limitations of using a spreadsheet become apparent when utilizing the provided data to generate the program and committee proposals, i.e., it is hard to check for duplicated names and overlaps in their current formats – an automation without manual intervention is currently not possible. Finally, a significant amount of manual work is required to format the data and input it to the ETAPS and Easychair websites.

The client wants two systems, each pertaining to the issues listed above. The importance of providing a front-end to enable the members to provide their information in a correct format was highlighted for both issues. In both cases the information should be processed automatically while still allowing the client to edit the proposed schedule and committees. Finally, the information provided to the webpages should be generated automatically, ideally also synchronize the information to the ETAP website.

The client provided the team with the following data:

1) Contact information to the ETAPS website administrator,
2) Two Spreadsheets with sample data for the program schedule,
3) Two Spreadsheets with sample data for the selection committees.

Finally, it was agreed that the team will reconvene with the client in the next meeting session after formulating the initial sets of requirements, providing an initial overview of the project and possible initial solutions. In the next meeting a first overview of the system will be provided and an initial timeline to design and implement the solutions such that a more formal project proposal can be agreed upon by both parties.

Meeting Report (Week 2 - 14.02.23)

Abbreviation	Name
System 1	Conference Scheduling System
System 2	Committee Seat System

Topic	Comments
Prior meeting report discussion	We reviewed the report from the last meeting (09.02.23) and confirmed the changes. The goal was to understand the workflow of the client concerning the two systems such that we can correctly illicit requirements.
Requirements document	We reviewed the requirements document that was prepared to discuss the correctness. The goal is to create a comprehensive requirements overview (with a list of requirements, some of which are subject to change after user-testing). The client expressed the following adaptions/requirements to the workflow: 1) System 1: The client creates a multi-conference outline/template manually, issuing timeslots to the individual conferences that they can fill in the data. 2) System 2: A template for proposing committee members will also be provided. 3) Systems 1 and 2 require an interface, such that the individual conference programs and conference committee seats data can be entered by the respective conference chairs for the slots that are assigned to them (in the created template from 1 and 2. 4) Once the data has been entered by the conferences, the client will verify the program of the multiple conferences and adjust where needed. Additional Requirements: 1) Browser Compatibility: Include Safari 2) Device Compatibility: a) Smartphones (not essential) b) Desktops/Laptops (essential) 3) Adjust “Max 5 Clicks” to: Provide templates with smart default selections that can be adjusted by the user if needed. 4) Easy to use fields: E.g. When there are papers with 8 authors, there should just be a single name field in which the names can be entered, separated by commas. (Instead of manually clicking multiple fields).

Project Proposal & Mock-ups	• Some of the wording, especially in the Risks section must be adapted. • Once finalized, the proposal will be sent to the client.
Meetings & Absences	• The client will not be available on Tuesday, 07.03.23. The meeting of that week is rescheduled to Monday, 06.03.23 at 17:00h.
Planning (Deliverables): Project proposal & Mock-up by next week	• The requirements document will be adjusted in accordance with the abovementioned changes. • The project proposal document will be adjusted in accordance with the abovementioned changes. • Initial mock-ups of the system(s) will be provided by the 16.02.23, such that initial feedback can be gathered, and adaptions made. There will be joint review session in the next meeting, Tuesday 21.02.23.

Meeting Report (Week 3 - 21.02.23)

Abbreviation	Name
System 1	Conference Scheduling System
System 2	Committee Seat System

Topic	Comments
Project Proposal	Overall, the project proposal in its current form lacks details, especially pertaining to: 1) Section “Project Organization”: Our choices should be motivated and elaborated, 2) Section “Responsibilities”: The task description should be elaborated, 3) Section “Planning”: A more comprehensive and descriptive overview should be provided, 4) Section “Planned Deliverables”: The requirements document should be incorporated into the proposal. The requirements should be explained and discussed, also a discussion on e.g., a) What is meant with duplicates (Different spelling of names, parallel sessions et cetera), b) What data format is expected as input, c) What is the solution to the issues at hand.
Requirements document	The requirements overview has some overlap between System 1 and System 2. These should be moved to the “Combined Systems” section. Additionally, the following requirements should be added/changed: 1) Requirement Nr. 5 (Modify): System 1 *must* let Users 1 and 2 correct possible typing mistakes after entering the data 2) System 1 (Add): System 1 *must* provide an overview of the number of assigned slots per conference. 3) System 1 (Add): System 1 *could* allow Users 2 to upload their Easychair html file containing a list of accepted papers and speakers, 4) System 1 (Add): System 1 *must* allow User 1 to create multiple parallel sessions for conferences individually. 5) System 2 (Add): System 2 *must* allow Users 1 and 2 to add new names, 6) System 2 (Add): System 2 *must* allow Users 2 to enter a reserve list of potential program committee members, 7) System 2 (Add): System 2 *must* automatically replace the slot of person A with person B from the reserve list once Users 2 change the status from “pending invite” to “declined invite”.
Further Questions	• Some conferences, e.g., TACAS have multiple double sessions happening in parallel. Since they are growing, there will be more parallel sessions occurring in the future. • The client decides when these parallel sessions occur. • Conferences have gaps/breaks in-between sessions.

Wireframe & Mock-up	While showing the Wireframe & Mock-up to the client, the following requirements came up: 1) System 1: When generating the program, it would be good to have the option to display plenary sessions. Some special sessions, e.g., Award sessions would not be assigned to a structure. 2) System 2: It would be nice to allow the conference chairs to upload CSV files containing their proposed committee data. 3) System 2: The client must be allowed to specify the number of people the conference chairs have to include in their program committees. 4) Systems 1 and 2: When displaying the data for the programs and committees it would be nice to have a list of potential issues that will be displayed to the client which can be manually verified.
Deliverables	We will provide an updated Project Proposal and Requirements overview for the client by the end of this week.
Meetings	Next Meeting: 06. March 2023

18.5 Appendix E: Data Inputs and Outputs

Inputs				
Type	**Requirement (Nr.)**	**Format, Input-method**	**Processing**	**Priority (MoSCoW)**
Speaker Data	RE6, RE23	User (manual) data entry, web form	The data will be transferred and formatted by the API and stored in the database.	Must
Speaker Data (Spreadsheet)	RE40	CSV file, upload field	The data will be stored in the database after processing the CSV file, predicting column names, and potentially requiring user feedback on how the columns of the file should be interpreted.	Could
Speaker Data (Accepted Papers)	RE12	Easychair HTML file, upload field	The data will be stored in the database after parsing the html file and reformatting it.	Could
Committee member Data	RE14, RE20, RE23	User (manual) data entry, web form	The data will be transferred and formatted by the API and stored in the database	Must
Committee member Data (Spreadsheet)	RE40, RE20	CSV file, upload field	The data will be stored in the database after processing the CSV file, predicting column names, and potentially requiring user feedback on how the columns of the file should be interpreted.	Could
Committee member Data (Accept / Reject invites)	RE18	User (manual) data entry, web form	The data in the backend will be updated according to the user inputs. By default, the status of the participants will either be: Pending, Accepted, Rejected	Must

Outputs				
Type	**Requirement (Nr.)**	**Format**	**Processing**	**Priority (MoSCoW)**
Conference program (Single)	RE1	Web app generated table	The data will be retrieved from the database by the front-end via API in the back-end.	Must
Conference program (Multiple)	RE7	Web app generated table		Must
Conference program problem highlights	RE8, RE9	Web app generated table	The data will be retrieved from the database by the front-end via API in the back-end. The data will have been processed by the back-end to highlight potential problems.	Must
Conference committee (Single)	RE23	Web app generated table	The data will be retrieved from the database by the front-end via API in the back-end.	Must
Conference committee (Multiple)	RE23	Web app generated table		Must
Conference committee problem highlights	RE15, RE16	Web app generated table	The data will be retrieved from the database by the front-end via API in the back-end. The data will have been processed by the back-end to highlight potential problems.	Must
ETAPS website conference program	RE10	YAML file	The conference program for the ETAPS website will be in the form of a serialized object in the specified (YAML) format and structure.	Should
Easychair invite format (Single)	RE17	Text file	The proposed committee overview for individual conferences will be in the form: FirstName LastName <E-Mail> and downloadable by the users in a text file.	Must

18.6 Appendix F: API Documentation

The API documentation is appended to this page.

API Reference

FastAPI

API Version: 0.1.0

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
HTTPBearer	http, bearer	

API
1. USERS

1.1 POST /users/register

Register

Register a new user with the following fields:

- **username**: Each user must have a username
- **email**: Each user must have a valid email
- **password**: Each user must have a password
- **role**: Each user must have either the role "admin" or "non-admin" depending on their assigned scope

The API will return the created user with a status-code = 201 on success

- NOTE: **this operation can only be performed by admin users**

REQUEST

REQUEST BODY - application/json

```
{
  _id       string
  username* string  3 to 15 chars
  email*    string
  password* string
  role*     enum    ALLOWED:non_admin, admin
                    An enumeration.
}
```

RESPONSE

STATUS CODE - 200: Register user

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type* string
  }]
}
```

1.2 POST /users/login

Login

Login with the following fields:

- **email**: a valid user email
- **password**: the corresponding user password

The API will return:

- **status-code = 200** on success
- **bearer access token**
- **user role**

REQUEST

REQUEST BODY - application/json

```
{
  email      string  DEFAULT:Ellipsis
  password*  string
}
```

RESPONSE

STATUS CODE - 200: Login user

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*   string
    type*  string
  }]
}
```

1.3 POST /users/logout

Logout

Logout a new user with the refresh token cookie:

- **refresh_token**: The cookie assigned to the client browser

The API will:

- return **status-code = 200** on success
- **revoke** the token
- **blacklist** the token

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

undefined

1.4 GET /users/refresh_token

Refresh Token

Refresh your refresh_token cookie:

- **refresh_token**: The cookie assigned to the client browser

The API will:

- return **status-code = 200** on success
- **refresh** your token cookie

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

undefined

1.5 GET /users/all

Get Users

Get all users, requires:

- **access_token**: In the request header,
- **refresh_token**: The cookie assigned to the client browser
- **this operation can only be performed by "admin" users**

The API will:

- return **status-code = 200** on success
- a list of **users**

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: List of users

RESPONSE MODEL - application/json

[{
Array of object:

_id	string		
username*	string	3 to 15 chars	
email*	string		
password*	string		
role*	enum		**ALLOWED:** non_admin, admin An enumeration.

}]

1.6 DELETE /users/{u_id}

Delete User By Id

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*u_id	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type* string
  }]
}
```

1.7 PATCH /users/{u_id}

Update User By Id

Get all users, requires:

- **access_token**: In the request header,
- **refresh_token**: The cookie assigned to the client browser
- **this operation can only be performed by "admin" users**

The API will:

- return **status-code = 200** on success
- the updated **user** object

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*u_id	string	

REQUEST BODY - application/json

```
{
  _id      string
  username string
  email    string
  password string
```

```
  role      enum    ALLOWED:non_admin, admin
                    An enumeration.
}
```

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
{
  _id       string
  username* string  3 to 15 chars
  email*    string
  password* string
  role*     enum    ALLOWED:non_admin, admin
                    An enumeration.
}
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type* string
  }]
}
```

2. CONFERENCES

2.1 GET /conferences/sessions/{c_id}

Get Session Conference By Id

Get a conference related to session creation by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

The API will return:

- the **conference** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*c_id	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
{
  _id          string
  year*        integer
  dates*       [string]
  committees*  [string]
}
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type* string
  }]
}
```

2.2 DELETE /conferences/sessions/{c_id}

Delete Session Conference By Id

Delete a conference related to session creation by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

The API will return:

- the **conference_id** of deleted **conference** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*c_id	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type*  string
  }]
}
```

2.3 PATCH /conferences/sessions/{c_id}

Update Session Conference By Id

Update a conference related to the session creation

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

Body:

- any fields that you want to update
- do **not** include the _id

The API will return:

- a list of updated **session_conference** objects
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*c_id	string	

REQUEST BODY - application/json

```
{
  year        integer
  dates       [string]
  committees  [string]
}
```

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
{
  _id          string
  year*        integer
  dates*       [string]
  committees*  [string]
}
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
     loc*
       ANY OF
       prop0
       string
       prop1
       integer
     msg*  string
     type* string
  }]
}
```

2.4 GET /conferences/sessions/

Get All Session Conferences

Get all conferences related to the session creation

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

- **year** by default the current year
- **conference_id** (optional) the conference id

The API will return:

- a list of **conference** objects
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
c_id	string	
year	integer	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
[{
Array of object:
   _id          string
   year*        integer
   dates*       [string]
   committees*  [string]
}]
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
   detail [{
   Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type* string
   }]
}
```

2.5 POST /conferences/sessions/

Create Session Conference

Create a conference related to the session creation

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

Body:

- **year** by default the current year
- **dates** a list of dates the conference will occur
- **committees* a list of participating committees

The API will return:

- a list of created **session_conference** objects
- **status-code = 200** on success

As can be seen below

REQUEST

REQUEST BODY - application/json

```
{
  _id          string
  year*        integer
  dates*       [string]
  committees*  [string]
}
```

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
[{
Array of object:
  _id          string
  year*        integer
  dates*       [string]
  committees*  [string]
}]
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*   string
    type*  string
  }]
}
```

2.6 GET /conferences/committees/{c_id}

Get Committee Conference By Id

Get a conference related to committee creation by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

The API will return:

- the **conference** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*c_id	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
{
  _id          string
  year*        integer
  dates*       [string]
  committees*  [string]
}
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type* string
  }]
}
```

2.7 DELETE /conferences/committees/{c_id}

Delete Committee Conference By Id

Delete a conference related to committee creation by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

The API will return:

- the **conference_id** of deleted **conference** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*c_id	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type*  string
  }]
}
```

2.8 PATCH /conferences/committees/{c_id}

Update Committee Conference By Id

Update a conference related to the committee creation

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

Body:

- any fields that you want to update
- do **not** include the _id

The API will return:

- a list of updated **committee_conference** objects
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*c_id	string	

REQUEST BODY - application/json

```
{
  year        integer
  dates       [string]
  committees  [string]
}
```

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
{
  _id          string
  year*        integer
  dates*       [string]
  committees*  [string]
}
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*   string
    type*  string
  }]
}
```

2.9 GET /conferences/committees/

Get All Committee Conferences

Get all conferences related to the committee creation

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **year** by default the current year
- **conference_id** (optional) the conference id

The API will return:

- a list of **conference** objects
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
c_id	string	
year	integer	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
[{
Array of object:
  _id          string
  year*        integer
  dates*       [string]
  committees*  [string]
}]
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*   string
    type*  string
  }]
}
```

2.10 POST /conferences/committees/

Create Committee Conference

Create a conference related to the committee creation

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

Body:

- **year** by default the current year
- **dates** a list of dates the conference will occur
- **committees* a list of participating committees

The API will return:

- a list of created **committee_conference** objects
- **status-code = 200** on success

As can be seen below

REQUEST

REQUEST BODY - application/json

```
{
  _id          string
  year*        integer
  dates*       [string]
  committees*  [string]
}
```

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
[{
Array of object:
  _id          string
  year*        integer
  dates*       [string]
  committees*  [string]
}]
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*   string
    type*  string
  }]
}
```

3. COMMITTEES

3.1 GET /committees/duplicate

Get Duplicate

Get all duplicates in the committees according to the parameters

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **year** (optional) the year of the conference
- **conference** (optional) the committee, e.g. "ETAPS"
- **conference_id** (optional) the conference id

The API will return:

- a list of **duplicate** objects
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
c_id	string	
conference	string	
year	string	

RESPONSE

STATUS CODE - 200: Get duplicate check by query

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type*  string
  }]
}
```

3.2 GET /committees/{c_id}

Get Committee By Id

Get a committee by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **committee_id** the committee id

The API will return:

- the **committee** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*c_id	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
{
  _id                          string
  conference*                  string
  year*                        integer
  slots*                       integer
  members [{
  Array of object:
     _id            string
     first_name     string
     last_name      string
     email*         string
     affiliation*   string  min:3 chars
     country*       string
     status*        enum    ALLOWED:ACCEPTED, REJECTED, PENDING, RESERVED
                            An enumeration.
  }]
}
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
     loc*
       ANY OF
       prop0
       string
       prop1
       integer
     msg*  string
     type* string
  }]
}
```

3.3 DELETE /committees/{c_id}

Delete Committee By Id

Delete a committee by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **committee_id** the committee id

The API will return:

- the id of the deleted **committee** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*c_id	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type*  string
  }]
}
```

3.4 PATCH /committees/{c_id}

Update Committee By Id

Update a committee by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

- **committee_id** the committee id

Body:

- any fields you want to update
- do **not** include the committee_id

The API will return:

- the updated **committee** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*c_id	string	

REQUEST BODY - application/json

```
{
  conference*                  string
  year*                        integer
  slots*                       integer
  members [{
  Array of object:
     _id           string
     first_name    string
     last_name     string
     email*        string
     affiliation*  string  min:3 chars
     country*      string
     status*       enum    ALLOWED:ACCEPTED, REJECTED, PENDING, RESERVED
                           An enumeration.
  }]
}
```

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
     loc*
       ANY OF
       prop0
       string
       prop1
       integer
     msg*   string
     type*  string
  }]
}
```

3.5 GET /committees/

Get All Committees

Get a list of committees by parameters

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **committee_id** (optional) the committee id
- **year** (optional) the year, by default the current year

The API will return:

- a list of **committee** objects
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
conference	string	
c_id	string	
year	integer	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
[{
Array of object:
   _id                          string
   conference*                  string
   year*                        integer
   slots*                       integer
   members [{
   Array of object:
      _id            string
      first_name     string
      last_name      string
      email*         string
      affiliation*   string  min:3 chars
      country*       string
      status*        enum    ALLOWED:ACCEPTED, REJECTED, PENDING, RESERVED
                             An enumeration.
   }]
}]
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
   detail [{
   Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type* string
   }]
}
```

3.6 POST /committees/

Create Committees

Create (multiple) committee(s)

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

Body (*a list of objects containing*):

- **conference** the committee this belongs to
- **year** the year
- **slots** total number of available slots
- **members** a list of members objects, by default can be empty

The API will return:

- a list of created **committee** objects
- **status-code = 200** on success

As can be seen below

REQUEST

REQUEST BODY - application/json

```
[{
Array of object:
   _id                          string
   conference*                  string
   year*                        integer
   slots*                       integer
   members [{
   Array of object:
      _id           string
      first_name    string
      last_name     string
      email*        string
      affiliation*  string  min:3 chars
      country*      string
      status*       enum    ALLOWED:ACCEPTED, REJECTED, PENDING, RESERVED
                            An enumeration.
   }]
}]
```

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
[{
Array of object:
   _id                          string
   conference*                  string
   year*                        integer
   slots*                       integer
   members [{
   Array of object:
      _id           string
      first_name    string
      last_name     string
      email*        string
      affiliation*  string  min:3 chars
      country*      string
      status*       enum    ALLOWED:ACCEPTED, REJECTED, PENDING, RESERVED
```

An enumeration.

```
  }]
}]
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type* string
  }]
}
```

3.7 PATCH /committees/{c_id}/{m_id}

Update Committee Member By Id

Update a committee member by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **committee_id** the committee id
- **member_id** the member id

Body:

- any member fields you want to update
- do **not** include the member_id

The API will return:

- the id of the updated **committee** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*c_id	string	
*m_id	string	

REQUEST BODY - application/json

```
{
  first_name   string
  last_name    string
  email        string
  affiliation  string
  country      string
  status       enum    ALLOWED:ACCEPTED, REJECTED, PENDING, RESERVED
                       An enumeration.
```

```
}
```

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type*  string
  }]
}
```

4. SESSIONS

4.1 GET /sessions/{s_id}

Get Session By Id

Get a session by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **session_id** the session id

The API will return:

- the **session** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*s_id	string	

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
full	boolean	

RESPONSE

STATUS CODE - 200: Get session by ID

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type*  string
  }]
}
```

4.2 DELETE /sessions/{s_id}

Delete Session By Id

Delete a session by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **session_id** the committee id

The API will return:

- the id of the deleted **session** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*s_id	string	

RESPONSE

STATUS CODE - 200: Delete Session by ID

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type*  string
  }]
}
```

4.3 PATCH /sessions/{s_id}

Update Session By Id

Update a session by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **committee_id** the committee id

Body:

- any fields you want to update
- do **not** include the session_id

The API will return:

- the updated **session** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*s_id	string	

REQUEST BODY - application/json

```
{
  title          string
  conference     string
  conference_id  string
  type           enum
                 ALLOWED:BREAK, CONFERENCE, SPEAKER, PLENARY, TUTORIAL, MISC
                 An enumeration.
  description    string
  session_order  integer
  day            string
  year           integer
  start_time     string
  end_time       string
  duration       integer
  presentations  [string]
}
```

RESPONSE

STATUS CODE - 200: Modified Session JSON

RESPONSE MODEL - application/json

```
{
  _id            string
  title          string
  conference     string
  conference_id  string
  type*          enum
                 ALLOWED:BREAK, CONFERENCE, SPEAKER, PLENARY, TUTORIAL, MISC
                 An enumeration.
  description    string
  session_order* integer
  day*           string
  year*          integer
  start_time*    string
  end_time*      string
  duration*      integer
  presentations  [string]
}
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
```

```
        string
        prop1
        integer
      msg*  string
      type*  string
  }]
}
```

4.4 GET /sessions/

Get All Sessions

Get a list of sessions by parameters

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **conference** (optional) the name of the conference that the session belongs to, e.g. "TACAS
- **conference_id** (optional) the conference that the session belongs to
- **s_type** (optional) the type of session, e.g. "BREAK", "CONFERENCE", "SPEAKER", "PLENARY", "TUTORIAL", "MISC"
- **session_order** (optional) the order of the session
- **day** (optional) the day of the session
- **year** (optional) the year, by default the current year
- **full** whether to return an array of nested objects or an array of references to IDs of programs, default = False

The API will return:

- a list of **session** objects matching the parameters
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
conference	string	
conference_id	string	
s_type	enum **ALLOWED:** BREAK, CONFERENCE, SPEAKER, PLENARY, TUTORIAL, MISC	
session_order	integer	
day	date	
year	integer	
full	boolean	

RESPONSE

STATUS CODE - 200: Get all sessions

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
```

```
    detail [{
    Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type* string
    }]
}
```

4.5 POST /sessions/

Create Sessions

Create (multiple) session(s)

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

Body (*a list of objects containing*):

- **title** (optional) the title of the session
- **conference** (optional) the conference this belongs to, e.g. "TACAS"
- **conference_id** (optional) the id of the conference it belongs to
- **type** the type of the session, e.g. "CONFERENCE"
- **description** (optional) the description of the session,
- **session_order** the order of the session, default = 1,
- **day** the date of the session
- **year** the year of the session
- **start_time** the start time of the session
- **end_time** the end time of the session
- **duration** the duration of the session in minutes
- **presentations** the list of presentation id's that were created, can be empty

The API will return:

- a list of created **session** objects
- **status-code = 200** on success

As can be seen below

REQUEST

REQUEST BODY - application/json

```
[{
Array of object:
  _id             string
  title           string
  conference      string
  conference_id   string
  type*           enum
                  ALLOWED:BREAK, CONFERENCE, SPEAKER, PLENARY, TUTORIAL, MISC
                  An enumeration.
  description     string
  session_order*  integer
  day*            string
  year*           integer
  start_time*     string
  end_time*       string
  duration*       integer
```

```
  presentations  [string]
}]
```

RESPONSE

STATUS CODE - 200: Created Session(s) with ID

RESPONSE MODEL - application/json

```
[{
```
Array of object:
```
  _id             string
  title           string
  conference      string
  conference_id   string
  type*           enum
                  ALLOWED:BREAK, CONFERENCE, SPEAKER, PLENARY, TUTORIAL, MISC
  description     string
  session_order*  integer
  day*            string
  year*           integer
  start_time*     string
  end_time*       string
  duration*       integer
  presentations   [string]
}]
```
An enumeration.

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
```
Array of object:
```
    loc*
      ANY OF
      prop0
        string
      prop1
        integer
    msg*  string
    type* string
  }]
}
```

5. PROGRAMS

5.1 GET /programs/

Get Program

Get a program by query

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

Request-query:

- **id** of the program
- **conference** (optional) that this program is associated with
- **session** (optional) that this program is associated with
- **session_id** (optional) of the session this program is associated with
- **title** (optional) of the program
- **authors** (optional) belonging to the program
- **year** (optional) of the program
- **day** (optional) of the program
- **type** (optional) of the program [deprecated]
- **start_time** (optional) of the program
- **end_time** (optional) of the program
- **duration** (optional) of the program in minutes

The API will return:

- a list of matching **program** objects
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
_id	string	
conference	string	
session	string	
session_id	string	
year	string	
title	string	
authors	string	
day	string	
type	string	
start_time	string	
end_time	string	
duration	string	

RESPONSE

STATUS CODE - 200: Get one or more programs

RESPONSE MODEL - application/json

```
[{
Array of object:
   _id           string
   conference*   string   max:20 chars
   session*      string
   session_id    string
   title*        string
   authors*      string
   year*         integer
   day*          string
   type          integer
   start_time    string
   end_time      string
   duration*     integer
}]
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
   detail [{
   Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type* string
   }]
}
```

5.2 POST /programs/

Create Program

Create new programs

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

Body (a list of Programs each containing):

- **conference** (optional) that this program is associated with
- **session** (optional) that this program is associated with
- **session_id** (optional) of the session this program is associated with
- **title** (optional) of the program
- **authors** (optional) belonging to the program
- **year** (optional) of the program
- **day** (optional) of the program
- **type** (optional) of the program [deprecated]
- **start_time** (optional) of the program
- **end_time** (optional) of the program
- **duration** (optional) of the program in minutes

The API will return:

- a list of created **program** objects
- **status-code = 201** on success

As can be seen below

REQUEST

REQUEST BODY - application/json

```
[{
Array of object:
   _id          string
   conference   string
   session      string
   session_id   string
   title        string
   authors      string
   year         integer
   day          string
   type         integer
   start_time   string
   end_time     string
   duration     integer
}]
```

RESPONSE

STATUS CODE - 200: Add one or more new programs

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
   detail [{
   Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type* string
   }]
}
```

5.3 DELETE /programs/

Delete Program

Delete a program by query

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

Request-query:

- **id** (optional) of the program
- **conference** (optional) that this program is associated with
- **session** (optional) that this program is associated with
- **session_id** (optional) of the session this program is associated with
- **title** (optional) of the program
- **authors** (optional) belonging to the program
- **year** (optional) of the program
- **day** (optional) of the program

- **type** (optional) of the program [deprecated]
- **start_time** (optional) of the program
- **end_time** (optional) of the program
- **duration** (optional) of the program in minutes

The API will return:

- a count of deleted objects
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
_id	string	
conference	string	
session	string	
session_id	string	
year	string	
title	string	
authors	string	
day	date	
type	string	
start_time	time	
end_time	time	
duration	string	

RESPONSE

STATUS CODE - 200: Delete one or more programs

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type*  string
  }]
}
```

5.4 GET /programs/duplicate

Get Duplicate

Get a list of duplicate objects by parameters

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **year** of the program
- **day** of the program

The API will return:

- a list of created **duplicate** objects
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*year	string	
*day	string	

RESPONSE

STATUS CODE - 200: Get duplicate check by query

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type*  string
  }]
}
```

5.5 PATCH /programs/update

Update Program

Update a program by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **id** of the program

Body:

- any fields you want to update
- do not include the **id** of the program in the body

The API will return:

- the updated **program**
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
_id	string	

REQUEST BODY - application/json

```
{
  _id         string
  conference  string
  session     string
  session_id  string
  title       string
  authors     string
  year        integer
  day         string
  type        integer
  start_time  string
  end_time    string
  duration    integer
}
```

RESPONSE

STATUS CODE - 200: Update a program by id

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*   string
      type*  string
  }]
}
```

6. EXPORTS

6.1 GET /exports/conference

Get Conference Yaml

Get a YAML object in the export format

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **year** (optional) the year, by default the current year

The API will return:

- a JSON structure according to the export format to be converted to YAML file in front-end
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
year	integer	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type* string
  }]
}
```

6.2 GET /exports/committee

Get Accepted Committee Csv

Get a string object in the export format

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **year** (optional) the year, by default the current year
- **conference** (optional) the conference

The API will return:

- a string object according to the export format
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
year	integer	
conference	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type*  string
  }]
}
```

7. ACCEPTED PAPERS

7.1 GET /accepted_papers/

Get All Accepted Papers

Get a list of accepted papers by parameters

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **conference** (optional) the conference, e.g. "TACAS"
- **year** (optional) the year, by default the current year

The API will return:

- a list of **accepted papers** objects
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
conference	string	
year	integer	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type*  string
  }]
}
```

7.2 POST /accepted_papers/

Post Accepted Papers

Upload the accepted papers HTML File

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **year** the year of the conference to upload this for
- **conference** the conference that this list belongs to, e.g. "TACAS"

File:

- Accepted Papers HTML File exported from the EasyChair website as Multipart/Form-data

The API will return:

- **string** = "inserted new papers"
- **status-code = 200** on success

As can be seen below

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*year	integer	
*conference	string	

FORM DATA PARAMETERS

NAME	TYPE	DESCRIPTION
file	string(binary)	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type*  string
  }]
}
```

7.3 DELETE /accepted_papers/{ap_id}

Delete Accepted Papers

Delete a list of accepted paper by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header
- **accepted_paper_id** the accepted paper list id

The API will return:

- the id of the deleted **accepted paper list** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*ap_id	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
      loc*
        ANY OF
        prop0
        string
        prop1
        integer
      msg*  string
      type*  string
  }]
}
```

7.4 DELETE /accepted_papers/{ap_id}/{p_id}

Delete Paper From Accepted Papers

Delete a paper from the list of accepted papers by id

Request:

- **refresh_token Cookie** must be included and valid
- **access_token** must be included in the header

- **accepted_paper_id** the accepted paper list id
- **paper id** the paper id

The API will return:

- the updated **accepted papers list** object
- **status-code = 200** on success

As can be seen below

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*ap_id	string	
*p_id	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
    Array of object:
    loc*
      ANY OF
      prop0
      string
      prop1
      integer
    msg*  string
    type*  string
  }]
}
```