RoyalTies

M11 Design Project Module (2023-1A)

UNIVERSITY OF TWENTE.

Design Project Report

M11 Design Project Module (2023-1A)

RoyalTies - Royalty software for music authors and music publishers

Students

David Galati	(s2539289)
Sebastian Mihai	(s2541475)
Radmehr Ghadiri	(s2452650)
Brianna Drîngă	(s2542072)
Michiel van Huijstee	(s2266156)

Supervising Professor

Dr. Ing. Moritz Hahn (e.m.hahn@utwente.nl)

RoyalTies Representative

Vanessa Maurischat (maurischat@kosmopolitmusic.de) RoyalTies, Kosmopolit Music Group, Berlin

UNIVERSITY OF TWENTE.

Contents

1. Introduction

This project aims to fulfill the digital needs of Kosmopolit Music Group – a German music publishing company that provides consultancy with regards to copyrights management with the national agency GEMA, a government-mandated collecting society and performance rights organization based in Germany. They act as an intermediary between the copyright owners, the performers or their representatives who want to do live performances of the copyrighted content.

1.1. Goals

This section will highlight the goal of our project. Although GEMA already has an easy-to-use web application, it might be intimidating for small artists or event organizers to collect the necessary data from third parties whom they interact with. Tracks need information about all performers, who may, or may not have a GEMA registration number, and same for the performances. A track list (or setlist) must be filled in with all the necessary information. That can be easily done if all performers are already registered by GEMA, but one might not know all the necessary information to achieve this on their own. This product aims to facilitate the processes that music publishers follow while working with clients. In our case, it aims to improve the transparency of the data flow between the music authors, performers, event organizers, the publisher, and GEMA.

1.2. Scope

This section will set the boundaries for which parts of the domain this project will cover, and which parts lie outside of the project scope. This involves creating a user-friendly platform to simplify the processes associated with copyright management and the collaboration between music authors, performers, event organizers, the publisher, and GEMA. Therefore, the scope can be defined solely by the communication between all the aforementioned involved parties, and anything happening once the data reaches GEMA is outside of the scope of our project. Organizational processes that might concern the event organizer, such as contact with venue representatives, are also outside of the scope.

The legal aspects of the German copyright law are also not tackled throughout this document. It is important to note that no part of this project should be considered legal advice.

1.3. Product Description

The Royal Ties Music Platform is a software-as-a-service (SaaS) solution tailored for the complex world of music performances and copyrights. It caters to live performers, music authors, event organizers, and the musical copyright framework. In a bilingual interface (German and English), performers can register tracks, view past performances, and plan future ones. Organizers can select, modify, and oversee track lists, while also assessing the royalty fees per event. Music authors benefit from features allowing them to register their tracks, recognize which authors are performing their tracks, and even compare their records with GEMA's through an innovative CSV comparison tool. Central to the platform is a payment system for handling royalty distributions, accompanied by user-friendly account management features. The platform also provides essential administrative tools for the publishing company's employees, allowing them to effortlessly manage author registrations, oversee performances, and handle billing.

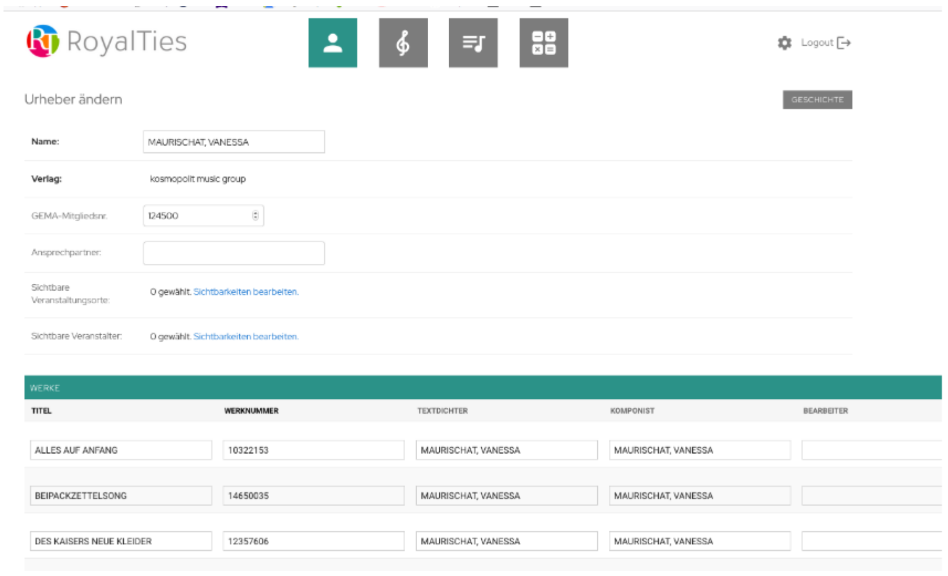
2. Domain

2.1. Current solution

The old layout of the application consisted of a Django interface that depicted multiple sections such as “Administration”, “Select artists to change”, “Add event” and so on. This follows the standard structure of CRUD operations. Each model instance is modifiable and depletable, and their layout is sometimes accompanied by filters.

The following screenshots depict the system from both the admin’s interface and the user's perspective. Such screenshots were provided by the client. A clear distinction between the actors and their available actions in the interfaces was not concretized.

Please refer to Appendix B: Initial RoyalTies for more illustrations about the old RoyalTies dashboard.



Urheber ändern

Name:	MAURISCHAT, VANESSA
Verlag:	kosmopolit music group
GEMA-Mitgliedsnr.	124500
Ansprechpartner:	
Sichtbare Veranstaltungsorte:	0 gewählt. Sichtbarkeiten bearbeiten.
Sichtbare Veranstalter:	0 gewählt. Sichtbarkeiten bearbeiten.

WERKE

TITEL	WERKNUMMER	TEXTDICHTER	KOMPONIST	BEARBEITER
ALLES AUF ANFANG	10322153	MAURISCHAT, VANESSA	MAURISCHAT, VANESSA	
BEIPACKZETTELSONG	14650035	MAURISCHAT, VANESSA	MAURISCHAT, VANESSA	
DES KAISERS NEUE KLEIDER	12357606	MAURISCHAT, VANESSA	MAURISCHAT, VANESSA	

Figure 1: “Urheber ändern” page (user perspective)

It was decided that to facilitate the new design requirements, the following changes are made:

1. **Header** tab: The upper section of the website will consist of a more design-friendly logo and a dynamic header which will compile together similar functionalities, following the most appropriate UX conventions (such as website personality, element shading, and accessibility concerns, etc.)
2. The "**Select performers to change**" section will be moved to its appropriate page within "Authors". This will allow users to access their respective songs easier, as it follows the pattern if index, view and edit transitions. Such convention advises that these changes can make navigation more intuitive and easier (Please refer to Appendix B: Initial RoyalTies)
3. Adding **songs** & **authors** sections will be updated with a more modern UI, and will be added accessibility-based components, such as "drag-and-drop" field for uploading documents.
4. The "**Select event for modification**" section will be moved towards the 'edit' step inside the index page of the events, following modern MVC architectural patterns. Same will follow for songs and authors.
5. **Administration**: The default Django administration component (resulted from the Django framework itself) will be positioned to the header and will lead to a separate page. This will make use of the grid pattern according to the modern User Experience guidelines.

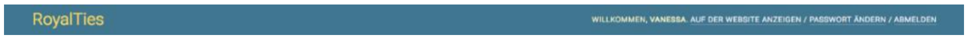

RoyalTies
WILLKOMMEN, VANESSA. AUF DER WEBSITE ANZEIGEN / PASSWORT ÄNDERN / ABMELDEN

Figure 2 Old website header

2.2. Market insights

The current solution involves GEMA, a royalty and copyright management organization in Germany, the equivalent of the United States' ASCAP or BMI [1], [2]. Currently, the authors of the song must list all concerts in the GEMA Dashboard manually, once a year, on January 1st, on a period that extends until December. GEMA will then process such data and provide a payment overview that results from performing the mentioned songs. Possible lack of data is detrimental to such a process, which is said to be around 20%, implication that will usually involve a complaint such that the losing party will recover their loss. This solution usually also involves head-to-head comparison against the performers' own system, for such dates to be identified and acted upon.

Currently in Germany, the performer can hold the rights and cannot be taken away from them using any compensation, as opposed to United States. However, this implies that it needs an entity or third party to process the royalties and submit them to GEMA.

Competitors

As of September 2023, there are no competitors in the market. The closest product that resembles our design the most is "LISTIG REWNu", introduced by DE-Parcon [3]. The official slogan of the cited product is "Aufführungen tracken, prüfen und reklamieren. Schnell und einfach zur Nachvergütung", which directly translates to ""Track, check and complain about performances. Quickly and easily for subsequent remuneration". When it comes to using author or song-related data, such a client provides an audit of the author's performances that have spanned the year. This is later stored and used to create an index of records that is contained within a similar format to this application, with selective headers, and which is directly related to the document that needs to be generated. Such data is later used to automatically correlate the input data with the GEMA form structure using AI algorithms, and moreover, the transactional state of such actions (store, complaint, etc.) is also trailed and available to the stakeholder [3].

Distinctive Features

1. DE-Parcon advertises that the end stakeholder of such performances, e.g., the producer/writer can easily be notified about the management of intellectual properties and the status of ongoing transactions.
2. The data generated from such algorithms will be analyzed head-to-head with the input data. Such algorithms are not clearly defined, yet it is bound to differ from the current approach of our application.
3. DE-Parcon advertises that multiple input structures or document types will be accepted by their algorithms [3].

Similar features

1. As clients centrally report to GEMA, each party who submits performances through their dashboard will have to comply with their structure. In the end, this implies that the data that is being collected by both organizations will have to match.
2. Both the client and the current application will be accessible for external use via third-party authorization or individual licensing. The use of API integrations will be further discussed in the upcoming Architectural Design section and resemble the approach of DE-Parcon, with regards to external authorization.
3. The application deployment and data storage platforms are both "cloud" based [3].

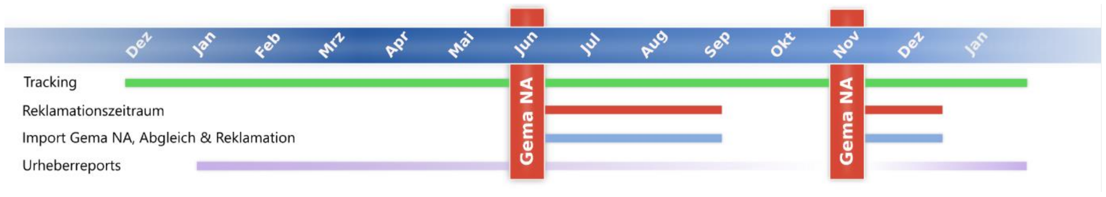


Figure 3 REWNu | DE-Parcon - Competitor's website main page [3]

Historical data

Historical data was provided to the development team at the beginning of the first design sprint. The document containing this data was aimed to monitor the performance statement and user data of artists.

Due to the high confidentiality of the document, such input data will not be shared throughout this document. The following headers were mentioned: YKTO, HKTO, HNAME, KOMPO, BEARB, TITEL, WERK, FAS, SEG, SPARTE, DATUM_VAN, DATUM_BIS, NUTZUNGSJAHR, BEGINN, ORT, VA-RAUM, VERANST, ABR, ANZAHL, MLEITER, INTERPRET, REFKZ. Each of these terms are further described in the 3.6 GEMA submission procedure section.

The structure of this provided document helped us reflect on the type of data needed for our dashboard, comparison procedures, and artist data aggregation. To take accessibility into consideration, some of the headers were either combined, removed or omitted. Furthermore, the structure of this document helped in designing the attributes for each class during database design and helped us reflect on the relation between entities (artist, song, venue, etc.) These design choices paved the way for internal design reflection processes, and each are further explained in the upcoming Architectural Design section.

3. Requirement specification and analysis

3.1. Communication issues from the client side

Throughout the development of this application, it was expected by all parties involved that detailed requirements and end goals were made clear from the beginning of the first design phase. The client was only available for one meeting, scheduled for September 8 (one hour meeting), making it impossible for us to conduct a traditional feedback loop. Afterwards, communication on the part of the client ceased gradually, with a period of multiple weeks with no response.

To compensate for the lack of requirements, it was deemed necessary that design reflection meetings needed to be set up. Such requirements have been assessed, reflected upon, and implemented by the development team. Each of these was a result of a clear design and reflection process and tried to predict the end goals of the organization and application itself, as accurately as possible. An extensive description of these processes is described in 0 Reflection Meetings.

3.2. Stakeholders

- Publishing company
- Live artist
- Author (copyright owning entity)
- GEMA
- Event organizer

3.3. Functional requirements

The system should be able to…

- …display a clear list of songs that were / are going to be played during a performance.

- …display for each song, its author, composer, lyricist, editor.

As a performer, I want to…

- …fill in a list of songs that I want to play at a certain date.
- …fill in a list of songs to put in a track list.
- …access the full track lists that I have registered in the past.
- …see which performances have been sent to GEMA.

As an event organizer, I want to…

- … receive track lists from authors.
- … check received track lists from authors.
- … edit information in the track lists made by authors.
- … if needed, to override a performer's track information or supplement it.
- … create track list from scratch.
- …see how much I need to pay in royalties, per performance.

As an author, I want to…

- … add a song.
    - Title
    - GEMA Werknummer
    - Writer
    - Composer
    - Editor
    - Duration
- … register a track's information so it can be then checked and sent to GEMA.
- … register information needed for GEMA.
- … easily fill in information about my copyrighted content.
- … press a button that would print the difference in the list of events – between our system and GEMA's (comparing the CSV files)
- … access a list of performers that have played / are going to play my copyrighted content.

As a publisher, I want to…

Admin:

- As an admin I want to see a list of registered authors
    - For each author, I want to be able to see and change their information
- As an admin I want to see a list of registered performances
    - For each performance, I want to see their
        - location
        - start date
        - end date
- As an admin, I want to search for a specific performer by their name.
- As an admin I want to see a list of track list- ***Musikfolgen***
    - list of tracks - author,
- As an admin I want to see a list of outstanding payment bills

General user:

- As a user, I want to reset my password.
- As a user, I want to edit my profile information.
- As a user, I want to be able to leave fields where I don't yet know the information empty such that someone else down the data chain can fill that in for me (i.e. GEMA Numbers, Track Numbers, etc.)
- As a performer, I want to create an account:
    - Email
    - Password
    - GEMA number if already a member
- As an organizer, I want to create an account.
    - Email
    - Password

3.4. Non-Functional Requirements

- Each page of the web app must load within 3 seconds.
- The availability of the system should be a minimum of 99.8%.
- The system must be able to handle 10,000 users at the same time without any degradation in performance.
- Implement multi-factor authentication for all user accounts.
- The application should support screen resolutions ranging from 320x480 to 3840x2160 without loss of functionality.
- The application should comply with the General Data Protection Regulation (GDPR) and any other relevant data protection laws in Germany.
- The application should support both German and English at launch, with the capability to add additional languages in the future.
- Daily automatic data backups should be performed, with the ability to restore data within the past 30 days.

3.5. Requirement prioritization

To provide a clear prioritization and partition of concerns, it was deemed appropriate to create a MoSCoW table, that would help us systematically tackle the requirement elicitation procedures.

A subset of requirements did not need prioritization, such as those regarding privacy and security, which were deemed crucial by design. The others that needed debate were divided into Must/Should/Could/Won't swim lanes. The ones under the "Must" swim lane were prioritized, closely followed by the following 2 swim lanes, namely "Should" and "Could".

These structural decisions were a direct result of reflection processes (and a limited set of initial requirements from the client), which spanned throughout the project. Each of these reflective and design decisions are documented and mentioned in the following sections.

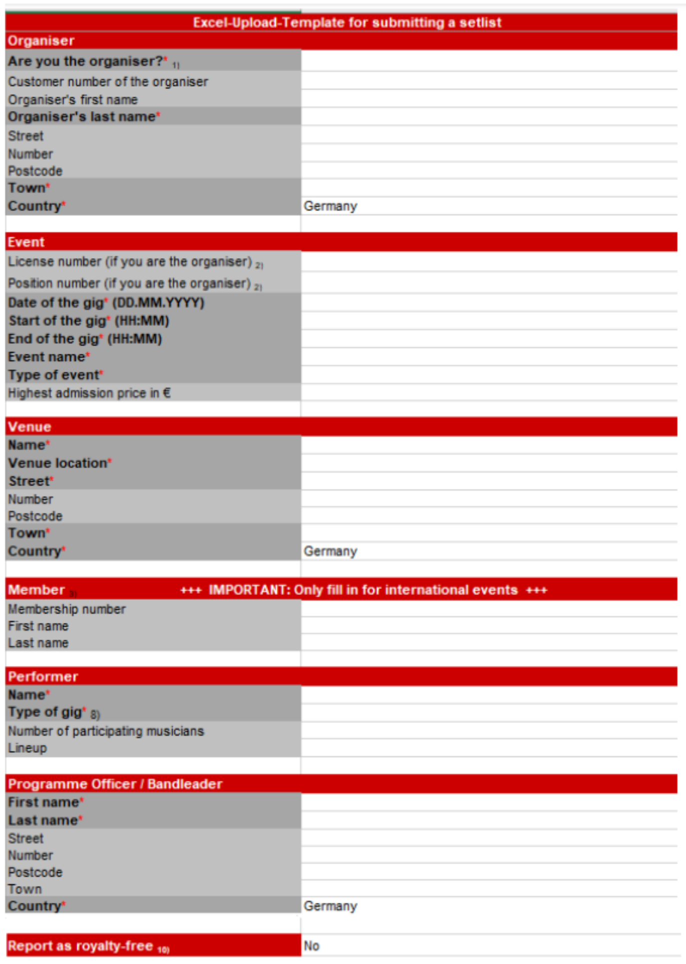
Table 1 MoSCoW Table

Must	Should	Could	Won't
• Basic overviews of • Performers • Authors • Event Organizers • Events • Tracks • Track lists • allow the admin to perform CRUD operations on the DB. • Working login system for admins, authors, and event organizers • ...display a clear list of songs that were / are going to be played during a performance. • As a live event organizer, I want... • to fill in a list of songs that were/ will be played at a certain event. • As an admin I want to see a list of registered authors • For each author, I want to be able to see their information. • For each author, I want to change their information. • As an admin I want to see a list of track list-Musikfolgen • As an admin, I want to see a list of registered performances. • For each performance, I want to see them ○ location ○ start date ○ end date	• As a user, I want to edit my profile information. • As a performer, I want to fill in a list of songs that I want to play on a certain date. • As an author, I want to create an account. • As an author, I want to register a song. • As an artist, I want to add a song. • As an organizer, I want to create an account. • see how much I need to pay in royalties, per performance. • A Way to reset the password in case of forgotten password using email. • press a button that would print the difference in the list of events – between our system and GEMA's (comparing the CSV files) • display for each song, its copyright owner • Be able to delete all the data of a client	• offer services in both German and English • Such CRUD actions should all be translated in both Dutch and English via a translation library. • 2FA or MFA for authentication • ...provide a CSV file with payment data for each user on January 1st of each year	• AWS Implementation

3.6. GEMA submission procedure

The following section will contain a description of fields and GEMA-related notions that are not publicly available and can only be accessed by registered artists. Due to the lack of communication from the client side with regard to these forms, the sources of the following mentioned documents cannot be specifically linked.

1. Client requirement quote: “Match all data submitted via the track list [4] template with the NA-named form (one such document name example would be “**NA_01.06.2022_NA_U-MUSIK-(<CODE>) _2022(<ORG_NAME>)**” at the push of a button to identify the concerts that have not been charged.” This requirement is based on the underlying mechanisms of GEMA. The organization compiles the yearly concerts, undergoes the billing procedure and publishes the document on June 1, the year after the initial concert list was received from the authors/publishers.
2. The submission is a baseline for royalties' accountability. Through the form provided by GEMA, authorized entities can provide the songs played throughout a single concert. Such a form cannot handle multiple concerts, or songs played throughout multiple venues.
3. In the future, such forms will be extended to contain multiple track lists of concerts, spanned through an entire month, via an API.
4. Form structure (The official upload template GEMA setlist document is only available for authors via the website [5], [6])
5. Individual Track List: An individual track list of information needs to be submitted, through a form, for every song. See Appendix F: GEMA form for more information about the structure of this form.



Excel-Upload-Template for submitting a setlist	
Organiser	
Are you the organiser?* 1)	
Customer number of the organiser	
Organiser's first name	
Organiser's last name*	
Street	
Number	
Postcode	
Town*	
Country*	Germany
Event	
License number (if you are the organiser) 2)	
Position number (if you are the organiser) 2)	
Date of the gig* (DD.MM.YYYY)	
Start of the gig* (HH:MM)	
End of the gig* (HH:MM)	
Event name*	
Type of event*	
Highest admission price in €	
Venue	
Name*	
Venue location*	
Street*	
Number	
Postcode	
Town*	
Country*	Germany
Member 3)	**+++ IMPORTANT: Only fill in for international events +++**
Membership number	
First name	
Last name	
Performer	
Name*	
Type of gig* 8)	
Number of participating musicians	
Lineup	
Programme Officer / Bandleader	
First name*	
Last name*	
Street	
Number	
Postcode	
Town	
Country*	Germany
Report as royalty-free 10)	No

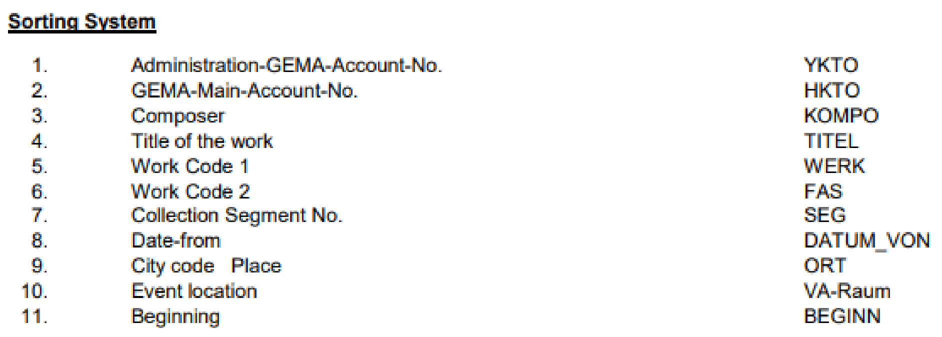
Figure 3 Track list general information form [3], [4]

6. Usage statement (GEMA-resulted CSV for the authors)
    a. GEMA responds to multiple structures of data, which are subsequently divided into multiple categories: E, U, FS, R, T FS. This application will focus on the historical data provided for **category U**. Such category is further split into 2 data formats: CSV formats old and new. The structure that the application uses is the "CSV Format new".
    b. The final data format structure: the source of the following information is the "**GEMA/IT Data Structure. Usage Data for Members with named Excel columns"** form, which is only available for registered artists [5], [6]. Overview of the sent track lists will be of the following form, usually in a CSV document (multiple formats available):
        i. SA: This term represents the category of the performed track
        ii. YKTO: This term represents the user's management GEMA ID. However, as of the current moment, GEMA does not provide third-party access or any API for such data transfer. Therefore, the task of filling in such information will be outsourced to the user.

iii. HKTO and HNAME: These two terms refer to the second type of account ("Account") "number" and "owner" respectively.
iv. KOMPO: Refers to the original author of the song
v. BEARB: This term refers to any modified version of the original song that was performed by the author. Current historical data does not provide any input for this field, yet it has been seen in the market that omitting this term can have detrimental aspects such as re-recording of an entire track should there be any copyright conflicts.
vi. TITEL: The title of the record.
vii. WERK and FAS – As cited by the official GEMA document "Work code 1" and "Work code 2" respectively
viii. SEG: This term represents the division number of the so-called income "collection". In the Euro currency, this ranges from 0 to 50 and can exceed 10 thousand, thus the segments are reported from 1 to 12 based on income. The term "Collection" specifically refers to social functions and not individual performances of the author.
ix. SPARTE: Refers to the type of music performed, ranging from "serious music" to "radio" or "commercial".
x. DATUM_VON and DATUM_BIS: Represents the date from and date to timestamps.
xi. NUTZUNGSJAHR: The year in which the song was performed.
xii. BEGINN: The starting time of the event.
xiii. ORT: The city location of the event.
xiv. VA_Raum: The venue location of the event.
xv. VERANST: The organizational entity of the event.
xvi. ABR: As mentioned in the documentation source specified, "Distribution (e.g., ABRE R 2006)".
xvii. ANZAHL: The amount of performed acts
xviii. MLEITER: The "musical director", empty if not applicable, and multiple values possible.
xix. INTERPRET: The artist, empty if not applicable, and multiple values possible.
xx. REFKZ: So-called "reference identifier", as specified in the documentation.

7. Sorting System

The sorting system that GEMA uses, and which the application follows. The following term is extracted from the "**Usage Data for Members**" form, which is private and can only be accessed by authors [5], [6].



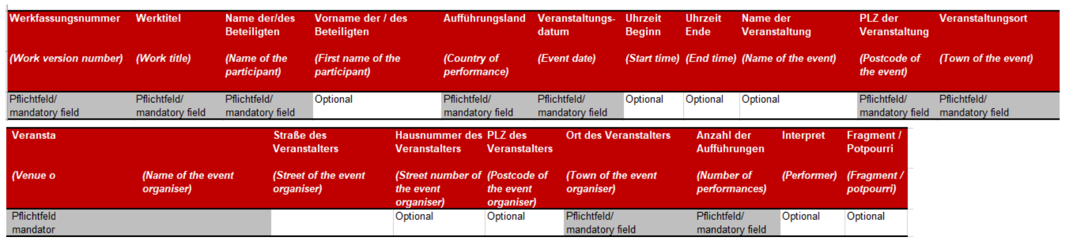
Sorting System

1.	Administration-GEMA-Account-No.		YKTO
2.	GEMA-Main-Account-No.		HKTO
3.	Composer		KOMPO
4.	Title of the work		TITEL
5.	Work Code 1		WERK
6.	Work Code 2		FAS
7.	Collection Segment No.		SEG
8.	Date-from		DATUM_VON
9.	City code	Place	ORT
10.	Event location		VA-Raum
11.	Beginning		BEGINN

Figure 4 Sorting system [3], [4]

8. GEMA complaints

a. Data of the author will be compared against the data received from GEMA, and by imposing the same structure, concerts that have not been identified and therefore not monetarily considered will be compiled and further reported back to GEMA.
b. This will be the second most important functionality of the dashboard and will follow all relevant ethical and privacy issues.
c. This functionality is only available through the **"GEMA claim form"**, and this step is only going to be performed after a successful comparison of the two data structures.
d. The previous step (of Data Submission) should be automated with the data checking process for missing events. Afterwards, the result should be automatically reformatted to the structure of GEMA complaint form.
e. Structure of the complaint form. The source of this form is the official **GEMA Template Live U 2020 Complaint Form**, once again restricted to artists only [7]–[9]



Werkfassungsnummer	Werktitel	Name der/des Beteiligten	Vorname der / des Beteiligten	Aufführungsland	Veranstaltungs-datum	Uhrzeit Beginn	Uhrzeit Ende	Name der Veranstaltung	PLZ der Veranstaltung	Veranstaltungsort
(Work version number)	*(Work title)*	*(Name of the participant)*	*(First name of the participant)*	*(Country of performance)*	*(Event date)*	*(Start time)*	*(End time)*	*(Name of the event)*	*(Postcode of the event)*	*(Town of the event)*
Pflichtfeld/ mandatory field	Pflichtfeld/ mandatory field	Pflichtfeld/ mandatory field	Optional	Pflichtfeld/ mandatory field	Pflichtfeld/ mandatory field	Optional	Optional	Optional	Pflichtfeld/ mandatory field	Pflichtfeld/ mandatory field

Veransta		Straße des Veranstalters	Hausnummer des Veranstalters	PLZ des Veranstalters	Ort des Veranstalters	Anzahl der Aufführungen	Interpret	Fragment / Potpourri
(Venue o	*(Name of the event organiser)*	*(Street of the event organiser)*	*(Street number of the event organiser)*	*(Postcode of the event organiser)*	*(Town of the event organiser)*	*(Number of performances)*	*(Performer)*	*(Fragment / potpourri)*
Pflichtfeld mandator			Optional	Optional	Pflichtfeld/ mandatory field	Pflichtfeld/ mandatory field	Optional	Optional

Figure 5 Complaint form [5]-[7]

4. Planning – Agile Methodology

Agile is an iterative project management approach where work is organized into short cycles called sprints. For the development of our project, we decided to implement this methodology due to its adaptability to change, as Agile embraces this throughout the project lifecycle and allows an iterative development that aligns with the needs of our project. This was especially helpful since the requirements evolved throughout the time frame of the project and prioritization shifts occurred.

Therefore, following the Agile iterative approach, we structured the 10 weeks corresponding to the timeline of the project into 5 sprints of 2 weeks each. The writing of the report started from the first week until the end of the project, since we decided to document all phases in the design trajectory as they were occurring, to ensure the accuracy of the design choices reflections.

A full overview of the Sprint planning involving the start and end date of each Sprint, along with its purpose, is highlighted by the table below:

Furthermore, a prioritization trajectory is reflected through the development process of the prototypes, further documented in the section.

Table 2 Project planning with Agile.

	Start date	End date	Description
Sprint 1	04/09/2023	15/09/2023	• Collection of functional and system requirements from the client • Technology Stack choices • Planning and management • Initial requirement elicitation
Sprint 2	18/09/2023	29/09/2023	• System design based on the requirements. • UI/UX design (no implementation) • Stubs of Frontend and Backend • MVP
Sprint 3	02/10/2023	13/10/2023	• User dashboard • Second prototype • Subscriptions management system
Sprint 4	16/10/2023	27/10/2023	• Final version • UI/UX design implementation • Payment system • Multilingual support
Sprint 5	30/10/2023	08/11/2023	• Testing • Bug fixes

4.1. Communication Medium(s)

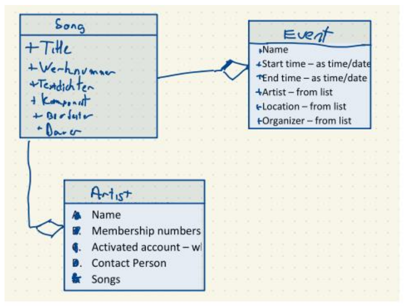
To ensure an efficient medium of communication and to schedule timely reflection meetings or deliverable deadlines, we have used Trello and GitHub. The meetings were planned one week in advance, and unanimously agreed on by all (or most) members. The topics of such meetings were in clear relation to the sprint goals, and meetings would be scheduled given the number of the remaining tasks left for that respective sprint. The meetings were mostly in-person, and exception cases were made for team members that could not attend in person. At the end of each week, the tasks scheduled on Trello were verified for completeness, and the following meetings were scheduled according to the result of that investigation.

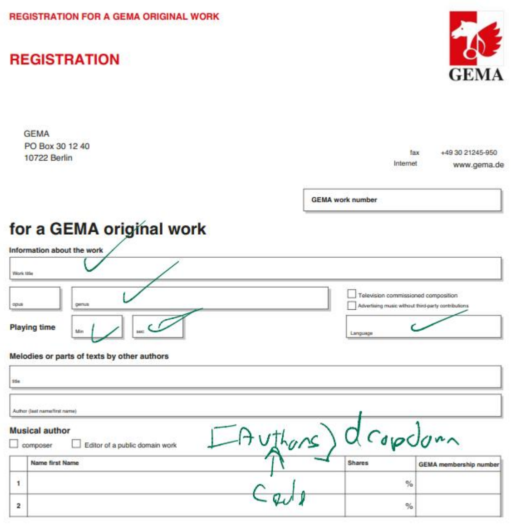
5. Architectural Design

In this section, a comprehensive analysis of the design choices will be showcased. Not only for designing the web application, but also external implications that influenced the creation of this product.

5.1. Database

The database has been built around the class diagrams that resulted from the requirement elicitation process. To determine the classes and agents, the GEMA submission form (available through the original GEMA work registration Dashboard [8]) was investigated, and attributes, methods, and classes were identified. For instance, fields such as 'title' and 'genre' were standalone attributes for the Track object, and multiple instances of Performer classes could be identified through their original author GEMA ID.




REGISTRATION FOR A GEMA ORIGINAL WORK

REGISTRATION

GEMA

GEMA
PO Box 30 12 40
10722 Berlin

fax +49 30 21245-950
Internet www.gema.de

GEMA work number

for a GEMA original work

Information about the work

Work title ✓

opus

genus ✓

☐ Television commissioned composition
☐ Advertising music without third-party contributions

Playing time Min ✓ sec ✓

Language ✓

Melodies or parts of texts by other authors

title

Author (last name/first name)

Musical author

☐ composer ☐ Editor of a public domain work

[Authors) dropdown

Cards

	Name first Name	Shares	GEMA membership number
1		%	
2		%	

Figure 6 Database design reflection and GEMA original work submission form [6]

5.2. Diagrams

Preliminary Class Diagrams

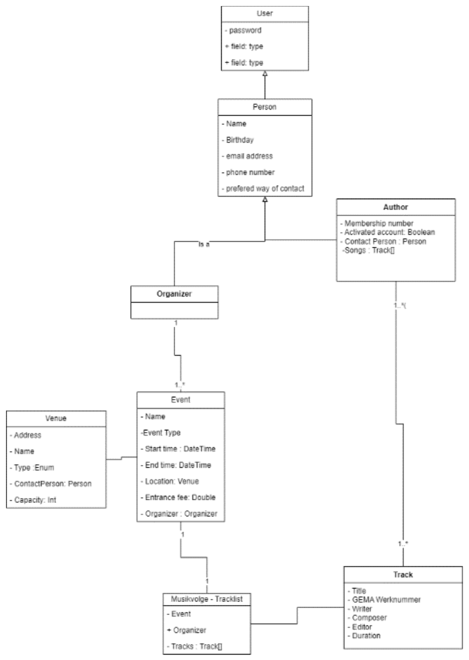


Figure 7 Class diagram

We initially used the publicly available information about GEMA, alongside the requirements written after the meeting with the client to sketch out the basic class structure of the application. This was very useful for use during the development process since it modeled the essential fields for each of the elements. For example, the Track had to have a Title, GEMA number, composer, editor, and duration. The track list was associated with an event, an event had an organizer and a venue, and so on. Further steps such as modeling the activity diagrams and the wireframes have been influenced by those design choices.

Class Diagrams

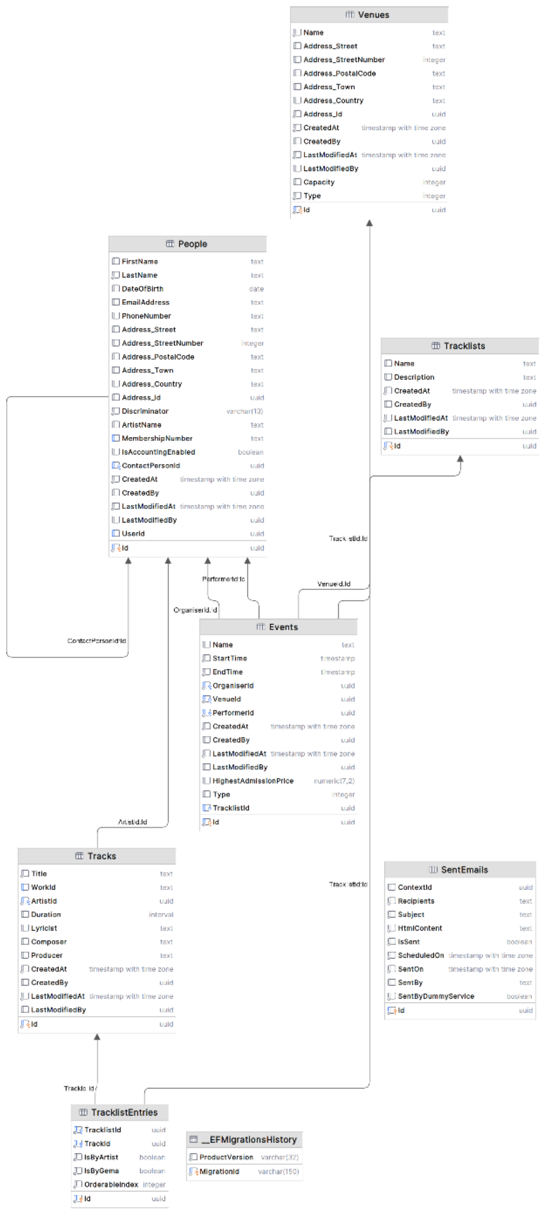


Figure 8 Class diagram

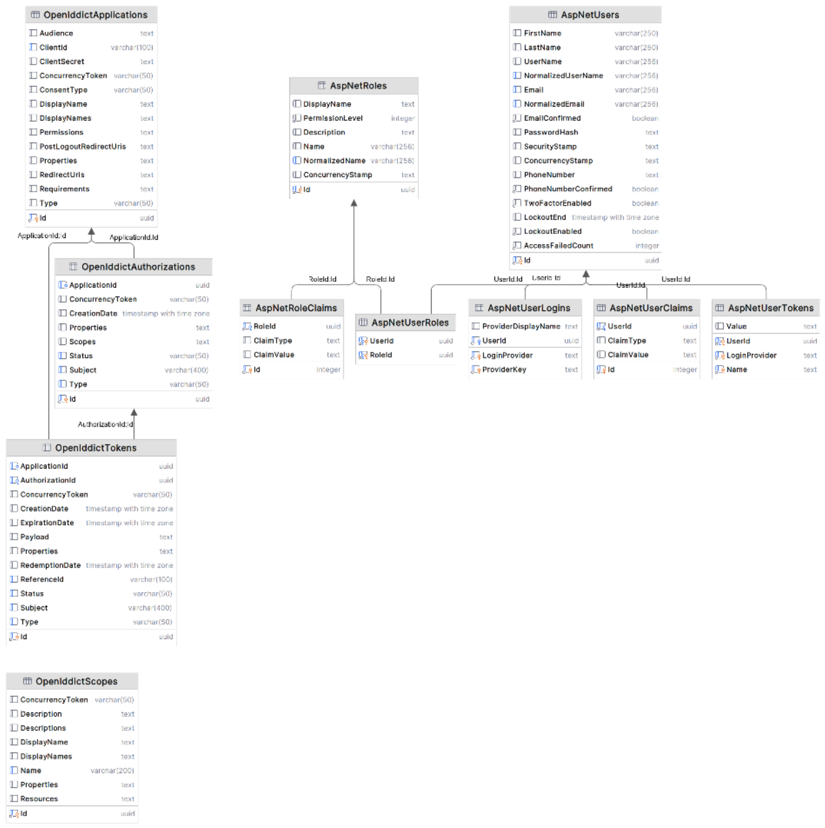


Figure 9 Class Diagram

Activity Diagrams

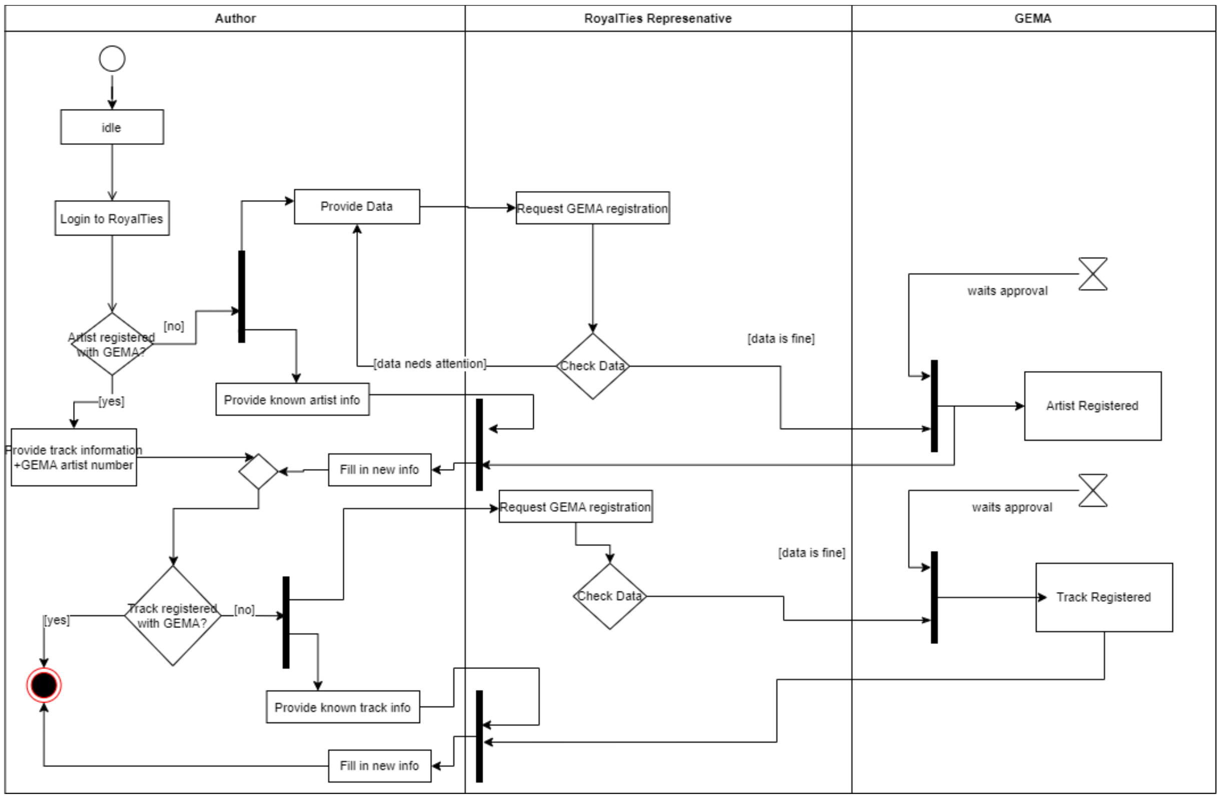


Figure 10 Activity diagram of the application

This activity diagram showcases how different actors' actions interact.

The three swim lanes represent how the stakeholders' business processes interact with the system. The first swim lane - the Author showcases the flow of actions depending on their current state.

If an author is not yet registered with GEMA, then their most probable first step would be to provide the data required for a GEMA registration. The Publisher representative will then check the data for completeness and correctness and pass them further to GEMA.

If the author is already registered with GEMA, then they can choose to register a new track. The publisher representative acts as a middleperson that would advise the author and facilitate a smooth registration with GEMA. By allowing incomplete data, this allows the user to fill in what information is available at the specific moment, and let the publisher gather the missing items. This is especially useful for performer collaborations and features, in which one performer might not have the other performers' information, or working with performers that might not even be registered to GEMA yet.

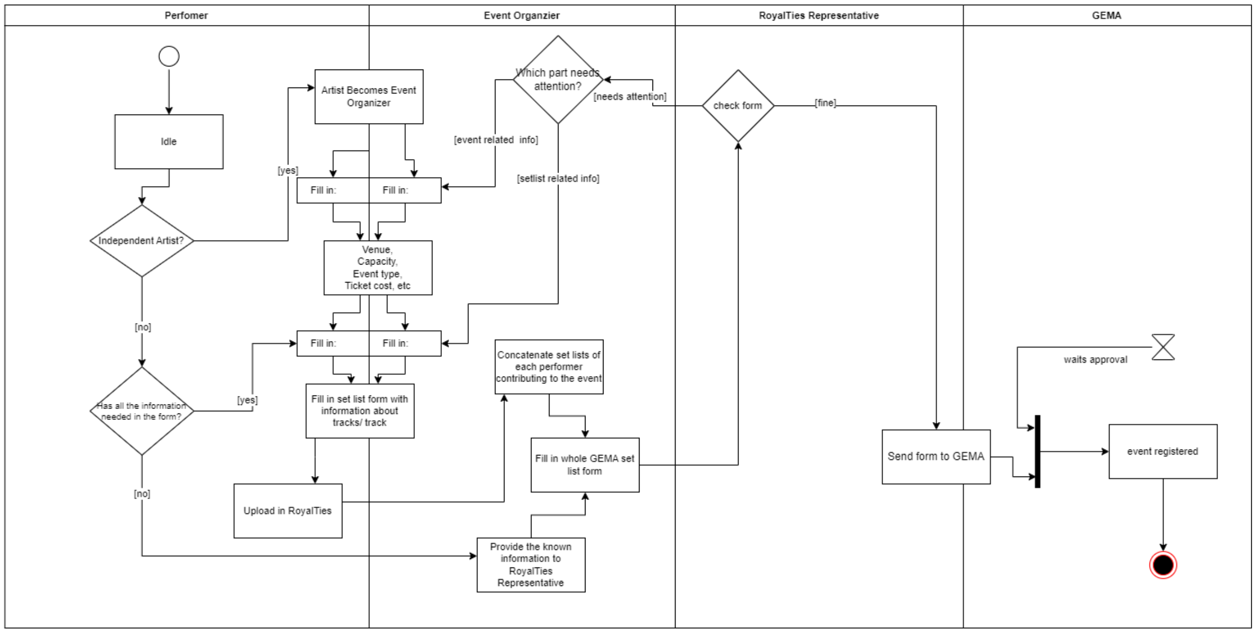

Perfomer
Event Organzier
RoyalTies Representative
GEMA
Idle
Independent Artist?
[yes]
[no]
Artist Becomes Event Organizer
Which part needs attention?
[needs attention]
[event related info]
[setlist related info]
Fill in:
Fill in:
Venue, Capacity, Event type, Ticket cost, etc
Fill in:
Fill in:
Has all the information needed in the form?
[yes]
[no]
Fill in set list form with information about tracks/ track
Upload in RoyalTies
Concatenate set lists of each performer contributing to the event
Fill in whole GEMA set list form
Provide the known information to RoyalTies Representative
check form
[fine]
Send form to GEMA
waits approval
event registered

Figure 11 Activity diagram showcasing the track list and event creation process

This Diagram showcases the activity flow for creating a track list and how the different actors take part in the business logic.

The natural flow goes from left to right, with each new actor adding their known information about the track, event, or venue.

In the case in which the event only has one performer, it would make sense that they could also take the role of the event organizer. In this case, the performer would add their own track details, the information about the venue, and send it to the Royal Ties representative, who would further check it and send the complete request to GEMA.

When there are larger events with multiple performers, the event organizer can either add the track list themselves, the information of the venue, info about each performer and track. The performers can put together a list of their tracks and send it to the event organizer which should compile them into the final list, which will be sent to a Royal Ties representative and then to GEMA.

Use-Case Diagrams

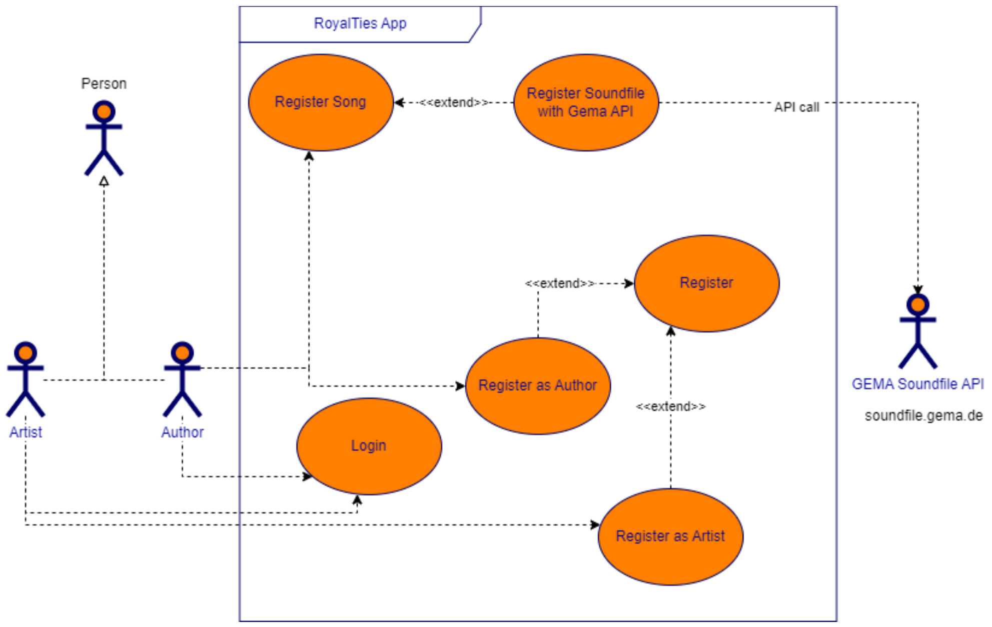


Figure 12 Use-case diagram for person, artist, author

This use case diagram highlights the distinct types of users, the main takeaway is that the registration page should be different for each of the users, but the login will stay the same.

GEMA has a well-documented public API for submitting such sound files [10]. This way, the authors can receive advice on the GEMA registration process and, at the same time, get help with registering their GEMA Sound file so that the copyright gets rightly associated to the track. This can also be further added to track lists by event organizers. GEMA Provides unique identifiers for authors, tracks, and such, as a GEMA number. That is given after the specific track is registered with GEMA.

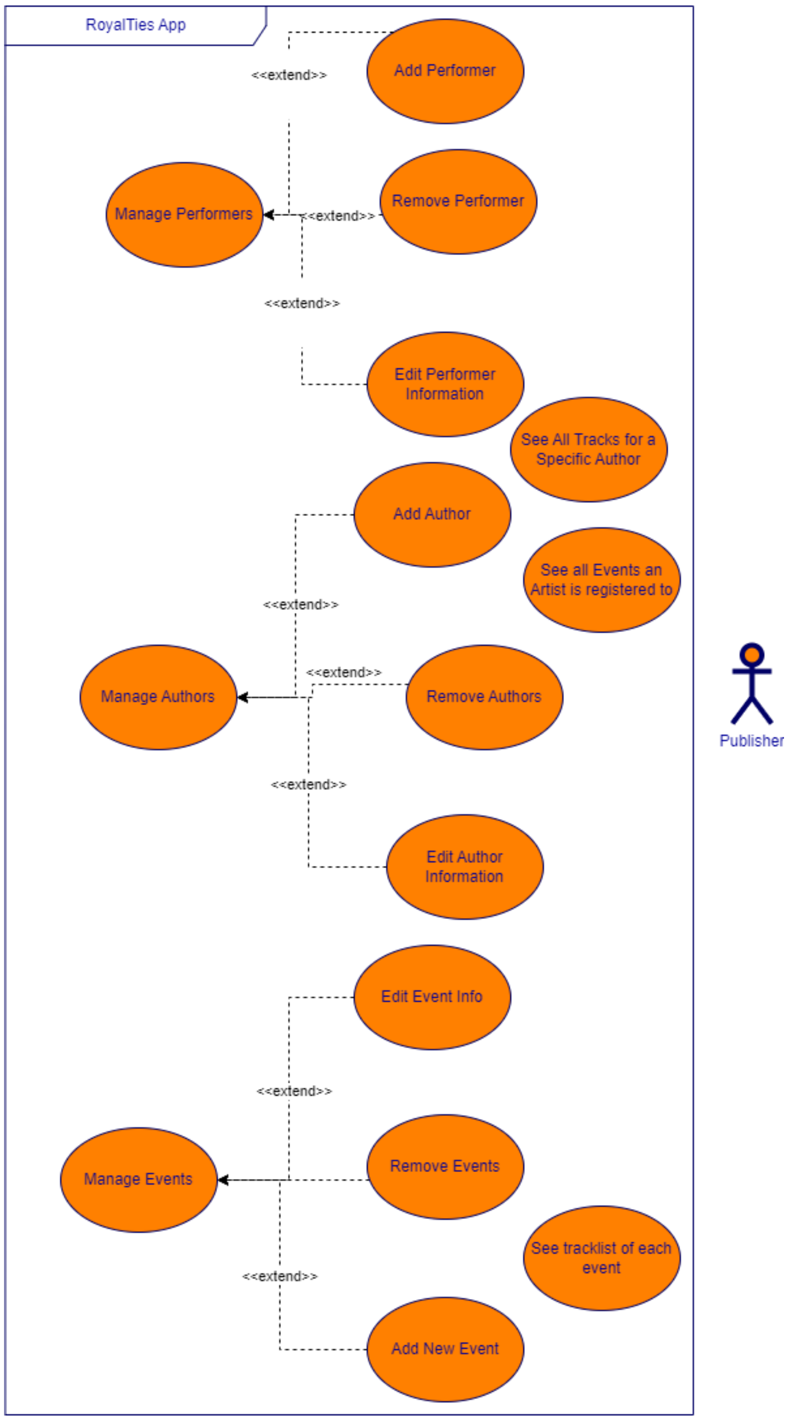


Figure 13 Use case diagram focusing on the publisher

The representatives of the public agency will use the online platform to perform different CRUD (Create, Read, Update, Delete) actions. One can think of this user as an administrator that has an overview of all the different entries such as – an author's tracks, a performer's performances, an event organizer's events, each with their track list.

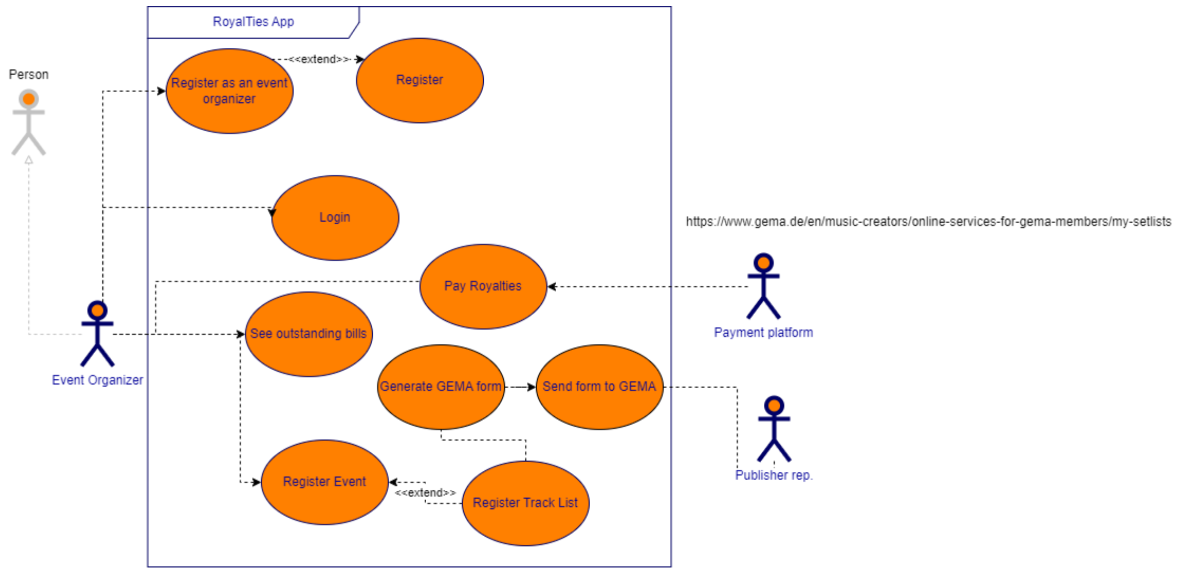


Figure 14 Use Case Diagram focusing on the Event Organizer

The event organizer would be the actor responsible for paying and registering the track lists to GEMA, GEMA provides a form [4]–[6] that covers features as:

- Tracks
- Information about organizer
- Information about the Location, Time, and Venue of the event

The event organizers would also need to pay the royalties. An overview of the due payments needs to be accessible from the web app, the list of events.

5.3. Tools

Database

PostgreSQL: After the initial reflection process, we decided on the specific actors that were going to interact with our application. This resulted in a set of attributes for each possible situation (Event Organizer, Author, etc.), and subsequently in classes for data-manipulated entities (Song, Event, Author, Form, etc.). As all these features were relationally associated, it was, given the requirements, most appropriate for us to use an RDBMS.

Backend

The backend is built using the ASP.NET Core framework. It is a solid state-of-the-art choice for web applications, given the prior experience of developers in this framework, we have decided to adopt this versatile tool. Moreover, the static typing of C# makes it a great choice, as opposed to other dynamically typed languages, with regards to maintainability and code understandability, a crucial aspect to consider regarding future development.

UI/UX Mockups

Balsamiq was used for UI/UX and diagram design, the team needed an interactive, accessible, and broad application. As our internal decision making revolved around diagrams and wireframes that Balsamiq is known for providing great support for, it was deemed appropriate by all team members.

Versioning

A GitHub repository was created for hosting the code and versioning services.

Organization and Task division

We have used Trello to divide and facilitate work for content distribution amongst the team members.

Hosting

Pineapp: As AWS was considered unfeasible given the design requirements reflection process, it was decided that the application will be hosted on Pineapp, owned by one of the team members, such that stakeholders could easily interact and test implemented functionalities, while the application was still under development.

Testing

Postman: In order to ensure that each page on the frontend side was functioning according to our expectations, endpoint testing, and system testing were deemed necessary. These tests were created given the metrics, which were subsequently introduced as a direct result of the functional and non-functional design choices.

Payment

For the payment mechanism, we decided to use Stripe. Such payment providers can ensure a mockup token and environment, without any need for actual payments, and can afterward be easily leveraged into a concrete mechanism. As it is one of the most renowned and safe alternatives, this tool will not involve security implications. As we were not given any payment information, due to privacy reasons, and due to its known compatibility and market presence, it was picked over the other available alternatives.

As opposed to its competitors, Stripe can provide a mock token for testing payments and authorized transactions, which can be proven especially useful in the development phases of the product.

5.4. Design Implications

- The target user of this application will be the music performers or publishers, as well as organizers or representatives of such performers. They will be able to conduct such automated processes themselves, without the current worry of compatibility or data loss.
    1. The current most important ethical implication is how the data is administered by the publisher on behalf of the performer.
    2. In United States for example, the entirety of the rights can be sold alongside the royalties.
- Another implication involves the organization of the actual concert or venue at which the song is being performed. In the current format, the organizer or publisher takes care of such implications. However, there needs to be mutual trust between such entities and the performer, such that the data is correct and unaltered at the point of submission. Using our application, such unbased trust does not need to exist anymore, as the authors can perform the GEMA upload actions themselves.
- Data handling is the most important privacy aspect of the application. The data will have to be stored, through the attributes of the decided class diagram, and later on distributed with GEMA. Such bridges of communication will therefore be detailed, and the client will be informed of any data processing and storage techniques. The user should be able to delete or edit their data at any point in time, as well as consent to any additional data storage such as essential cookies or marketing-related data such as Google Insights or data layers exported through third parties. Such data manipulation processes will be detailed on a separate page of the application, referring to the privacy statement, which includes the aforementioned privacy elements.
- One of the main aims of the application is to allow performers to register their tracks, keep track of payments and submit complaints, in an automatic manner, without the need for publishers, managers or other third parties.
- The application will function as a Software-as-a-Service and can directly function as a "publishing service" directly for the performer, and through monetizing such automated process, the organizational business needs would be met.

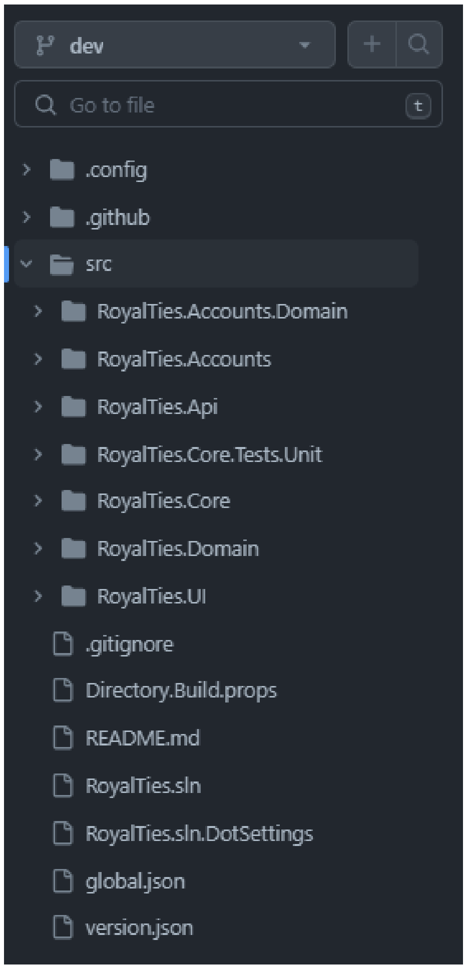
5.5. Back-End

The backend consists of two applications. The main application provides all business-related functionality, provides all data-related API endpoints and performs all tasks related to this data. The other application is a standard implementation of an OpenID Connect server to provide identity and authentication features to the main application.

Both applications are built on the newest release, version 8.0 (Release Candidate 2), of the ASP.NET Core framework available at the time of development. This high-performance, flexible, crossplatform framework provides many first-party tools that support in building high quality modern web applications.

Dependency Injection is used heavily in both applications, which allows us to follow the principle of Inversion of Control. This allows the components to be loosely coupled and easily testable separately, among other benefits.

Entity Framework Core is utilized as an ORM in both applications, to support fast, but also safe, development of the application, without having to worry about writing and updating SQL queries for every feature. Additionally, it provides the ability to automatically and dynamically alter entire database queries based on client request parameters.



```
> .config
> .github
v src
  > RoyalTies.Accounts.Domain
  > RoyalTies.Accounts
  > RoyalTies.Api
  > RoyalTies.Core.Tests.Unit
  > RoyalTies.Core
  > RoyalTies.Domain
  > RoyalTies.UI
  .gitignore
  Directory.Build.props
  README.md
  RoyalTies.sln
  RoyalTies.sln.DotSettings
  global.json
  version.json
```

Figure 15 Source code directory

Accounts Application

The main application requires authentication, and to support future expansion, it was important to perform authentication using bearer tokens instead of simple cookie authentication.

The RoyalTies.Accounts application, as it is called, is a standard OpenID Connect implementation using the open-source library OpenIddict. This application can be easily swapped for an external OpenID Connect provider such as Auth0 or Okta if the client is willing to pay for these services.

Creating and hosting this custom application instead of using a free external service allowed us to provide a cohesive experience with a consistent design across applications. This application is largely adapted from a template provided by OpenIddict, with account management pages being generated by a Microsoft-provided tool.

Main Application

To facilitate client interaction with the backend, the main application exposes a GraphQL endpoint. This endpoint implemented using HotChocolate's open source GraphQL server library allows quick and painless creation of various endpoints for all required data and actions for the client.

An important benefit of using GraphQL instead of standard REST endpoints, is that the client can explicitly request only the data that it needs at a certain point, and the GraphQL implementation allows automatic rewriting of database queries to only fetch that requested data, instead of entire database entities. It also allows the combining of multiple data requests in the same HTTP request, which further improves performance.

Architecturally, the main application is divided up into 4 different .NET projects.

This separation is mostly based on the principle of Clean Architecture, and while this application does not fully adhere to Clean Architecture it still allows for better maintainability, testability and scalability, among others [11].

- RoyalTies.Api
    - Implements GraphQL to facilitate communication with clients.
    - Provides models only used as Data Transfer Objects.
- RoyalTies.Core
    - Provides services and data access handling.
- RoyalTies.Domain
    - Host types that should be accessible by all applications that handle business actions, such as models, constants, and certain interfaces.
- RoyalTies.Core.Tests.Unit
    - Home to unit tests related to services in RoyalTies.Core
- RoyalTies.Core.Tests.Integration
    - Home to integration tests related to services in RoyalTies.Core

5.6. Front-End

Page description

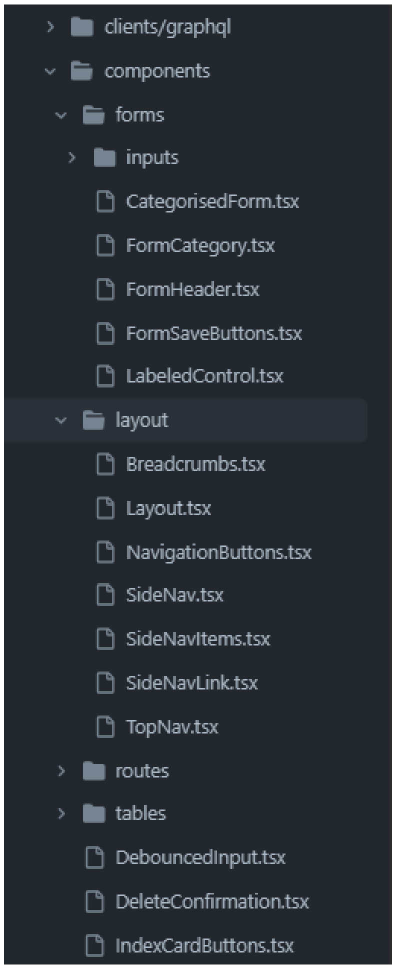
The following sections describe the UI elements for each of the web application dashboard pages, viewed through the perspective of an admin account.

Please refer to the section Appendix E: New UI for detailed screenshots of each of the following pages, as well as accessibility features such as day/night mode or automatic translations of each component.

- Login/Register page: The first page that a new user interacts with. They will be prompted with an email/password form modal, which also contain a “Remember me” checkbox and “Forgot your password?” and “Resend email confirmation” references. A user is also able to login via Microsoft or Google accounts, using OAuth2. The registration page contains an email/password/confirm password, detailed in the same way the login page is.
- Events page: The landing page of the application. The user is prompted with a table-like dashboard, to which they can add an event, using name, start time and location.
- Artist's page: Through this page, the user can search or add artists, using the name and number of tracks. Each of these artists requires a valid GEMA ID.
- Venues page: The location where the track is performed can be added to this page, using name and city.
- Compare page: The automatic and novel tool of the application, the CSV comparison can be achieved by uploading 2 CSV files, or one CSV file compared to the database, with the track lists via drag and drop or click. The result of this action will result in an excel file, with the differences that will indicate to the user the missing or unpaid tracks.
- Account management: This page includes multiple subsections, relevant to account data manipulation.
    - Profile: Through this settings page, any user can change their name and phone number (or add it if this has not been done yet)
    - Email: This section allows the user to edit their email.
    - Password: Through this page, the users can change their password, via 3 fields: Current password, the new one, and a confirmation field for the new password.
    - External logins: This page allows users to modify the third-party access to their account (namely Microsoft and Google tokens, used for OAuth2 authentication).
    - Two-factor authentication: The 2FA access can be set up or removed via this page. 2FA is not required for any user by default.
    - Personal data: The users cannot delete all their related data, but only their access to the webpage. This webpage is only providing the interface for the following actions: to comply with GDPR constraints, users can download or delete their data via this page, through the click of a button.
- Accessibility dropdown: The interface of the application can be changed via switching between day/night mode, choosing a language (that automatically translate each component) between Dutch and English, or sign out.

Components

Each frontend UI component reflects a design choice. They were implemented using TSX and follow the UX considerations mentioned above.



```
> clients/graphql
v components
  v forms
    > inputs
      CategorisedForm.tsx
      FormCategory.tsx
      FormHeader.tsx
      FormSaveButtons.tsx
      LabeledControl.tsx
  v layout
      Breadcrumbs.tsx
      Layout.tsx
      NavigationButtons.tsx
      SideNav.tsx
      SideNavItems.tsx
      SideNavLink.tsx
      TopNav.tsx
  > routes
  > tables
    DebouncedInput.tsx
    DeleteConfirmation.tsx
    IndexCardButtons.tsx
```

Figure 16 Component Hierarchy

Technical processes

The web app is built as a Single Page App using the ReactJS library. While the app mainly uses custom-made components, it also utilizes some premade components courtesy of Ant Design.

The app is styled using a custom-made Sass design library. This library also supports the functionality of defining multiple themes.

Authentication is handled based on the OpenID Connect specification. As part of this process, the user is redirected to a separate domain where the above-mentioned RoyalTies.Accounts pages are hosted to sign in, before then being redirected back to the main RoyalTies webapp. The frontend can then store JWTs to authenticate with the backend, and also use the metadata within the JWT to determine what kind of functionalities to show to the user. For example, the JWT contains the user's account type, which is used to determine whether to show the artist, event organizer, or admin interface to the user.

To support varied preferred languages, the front end was built with the idea of localization in mind. Every piece of text has been made localizable, so that it can be easily translated to any language required.

To facilitate communication between the frontend and the backend, GraphQL was used. This querying language allows for quick turnaround times in prototyping and development, as well as for fast requests in production, by easily allowing only the required data to be sent to the client.

Additionally, to further improve loading times, the app provides a service worker, which enables advanced client-side caching functionality. This allows for ensuring the static data of the app need not be fetched every single time a user opens or refreshes the webapp.

Finally, the webapp exposes a Progressive WebApp manifest, to allow users to install the webapp and make it effectively feel as though it is a native desktop application.

5.7. UI

To create the best possible experience and facilitate a straightforward and unambiguous user-application interaction, the following characteristics will be taken into consideration during front-end design.

a. The UI elements will either be custom components using React, TypeScript and Sass, or be premade components from the Ant Design React library that have been modified to visually fit in with the custom components.
b. The theme of the website will be consistent, from the first background page of the login page to the styling of the elements.
c. The navigation will be achieved via React Router, and the pages will be up for crawling.
d. For future development, such elements could use shading and depth, such as primary and secondary actions will be clearly distinguishable.

The result of such considerations is directly reflected in the new UI. Please refer to the section 5.6 Front-End, for an overview of said pages.

5.8. Accessibility & UX

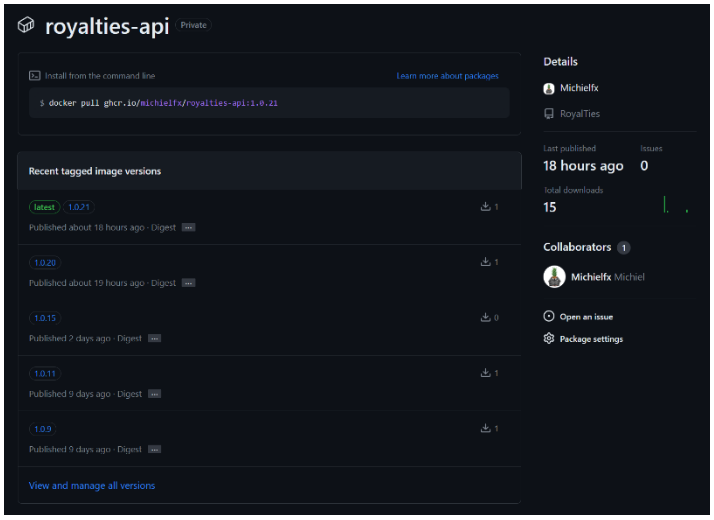
The automatic process of royalties and copyright administration of individual performers is one of the most crucial aspects of our application. This could lead, in the end, to highly abstracted or technically heavy notations and components, which can create a suboptimal experience for the end user. Therefore, the following reflections are meant to deal with this issue, aiming to deliver the best possible accessibility features and user experience.

1. Components
    a. The illustration will be created via Balsamiq, which is a software that allows outsourcing of UI illustrations to frontend frameworks for seamless incorporation. Such a dashboard will also allow us to validate UX implications, which will be further detailed in the upcoming section.
2. Implications (Ethical, user affect)
    a. First and foremost, the most important aspect of UX is the "site personality". This is achieved via requirements gathering and will determine how the users will feel about the website. The most important aspect of the application is automation and data privacy. Therefore, a more lean and solid color theme will be used, as well as more rigid structure to enrich the features we want to provide.
    b. As already mentioned, data privacy is one of the main concerns within the application. Therefore, any feature that involves or alters such processes will receive a primary color and accent, which will emphasize this.

c. Accessibility is another crucial aspect of the interface, as people with visual, auditory, or any other type of impartiality should be able to easily navigate through the website. This involves scaling out the images and text, labeling the fields and images, and considering the contrast rules of 5:1 to disseminate the color blindness effects.
d. Personal data manipulation actions (add, modify or delete data) will be clear and self-explanatory.
e. The privacy policy and GDPR sections will be clearly visible to the user, such that they are aware of such specific implications with regards to their data.
f. The application will not track the actions of the user.
g. The application will not save unnecessary data, besides the one that needs to be captured for basic functionality.
h. The application will not include intrusive cookies.

5.9. Deployment

a. The application is proxied by Cloudflare for faster and more secure deployment, and easier access to node servers regardless of geolocation. Additionally, CDN caching is also available for a faster edge-node bundle and static file caching, such that the UI components and overall dashboard provide a very low load-time for end users.
b. Hosting: To facilitate easier testing and accessibility to our prototypes, we have decided to host the application online. Moreover, we took this decision to catch errors that were not considered during development such as database connectivity errors or missing pages. The application was first deployed at: https://royalties.pineapp.nl/ . The process was achieved through the GitHub container registry, using the packages showcased below. The Docker images are currently built using GitHub actions, resulting in afferent Docker images.


royalties-api Private

Install from the command line

Learn more about packages

```
$ docker pull ghcr.io/michielfx/royalties-api:1.0.21
```

Recent tagged image versions

latest 1.0.21 — 1
Published about 18 hours ago · Digest

1.0.20 — 1
Published about 19 hours ago · Digest

1.0.15 — 0
Published 2 days ago · Digest

1.0.11 — 1
Published 9 days ago · Digest

1.0.9 — 1
Published 9 days ago · Digest

View and manage all versions

Details
Michielfx
RoyalTies

Last published: 18 hours ago
Issues: 0
Total downloads: 15

Collaborators 1
Michielfx Michiel

Open an issue
Package settings

Figure 17 royalties-api package in the GitHub Container Registry

c. Docker: To facilitate easier deployment and to ensure our current application was transferrable with the appropriate dependencies and environment, we have decided to upload the most recent version of the application to a Private Docker repository. The corresponding link and/or access will be given to the client at the end of the last development period.

6. Reflection on Design Choices

Throughout the development phase of the project, we have scheduled 2-4 internal meetings every week. One meeting was predestined to be a scrum meeting, whereas the other had the objective of corroborating multiple viewpoints with regards to requirements, design, and future shape of the project. This section provides the highlights of several of such meetings:

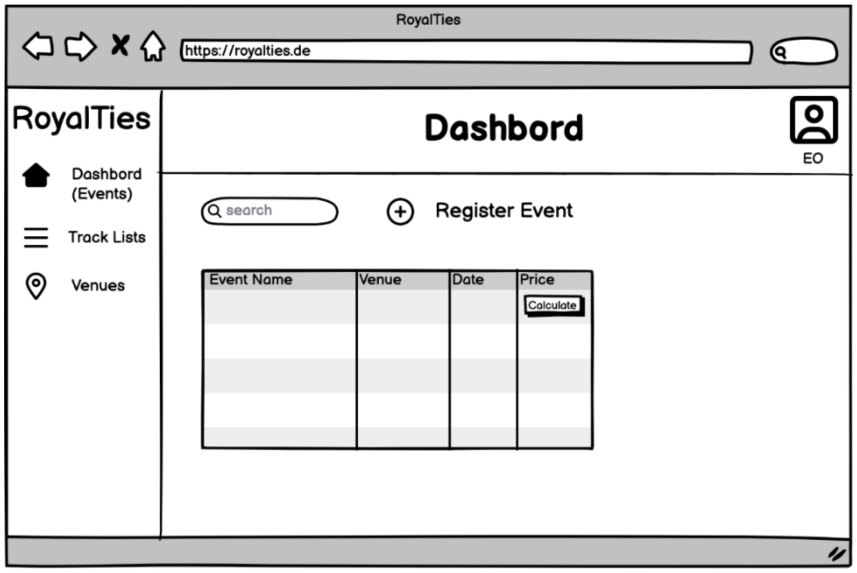


Figure 18 Initial Balsamiq dashboard

6.1. Proposal reflection

To validate the progress and the requirements that were compiled by the team, the client was sent a proposal, which acted as a stub of the final product.

This was achieved only after the requirements that were deemed crucial were agreed upon, designed and implemented. A subset of such requirements was:

- Clear distinction between performer, event organizer, author and Kosmopolitmusic employee, their responsibilities and the authorized actions in the dashboard
- A clear workflow of track list upload (data not persisted for the proposal)
- A clear distinction of the admin interface and its functions
- Basic authentication mechanism

The action of compiling the proposal has revealed multiple gaps and aspects where the design process can be improved, a result which has led us to pick up the MoSCoW structure for the future prioritization requirements.

As the distinction between actors was not set in stone, and no further information was provided in this sense at the time of reflection, we have also identified the need of creating a “Use-Cases Actor table”. See Appendix C: Use Cases-Actor mapping which aims to outline the workflow of each aforementioned actor throughout the dashboard.

Another reflection process that took part was to leverage the usage of multiple diagrams such as use case, activity, workflow or class diagrams.

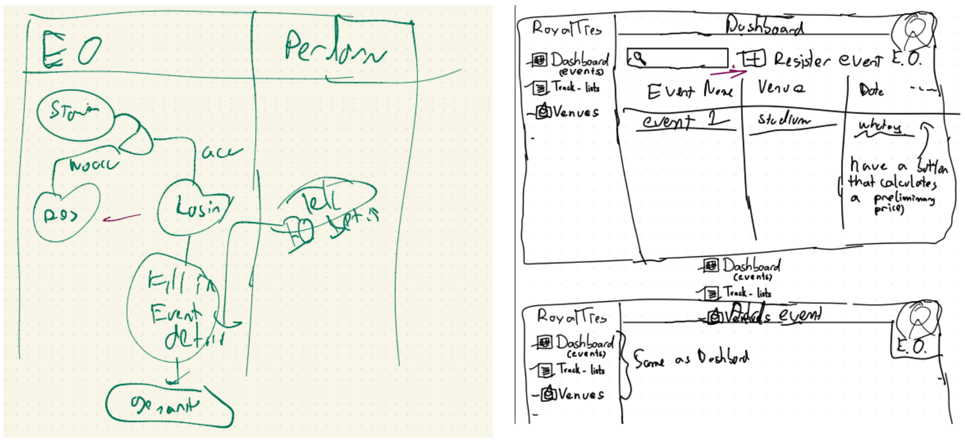


Figure 19 Wireframe diagram result via reflection process

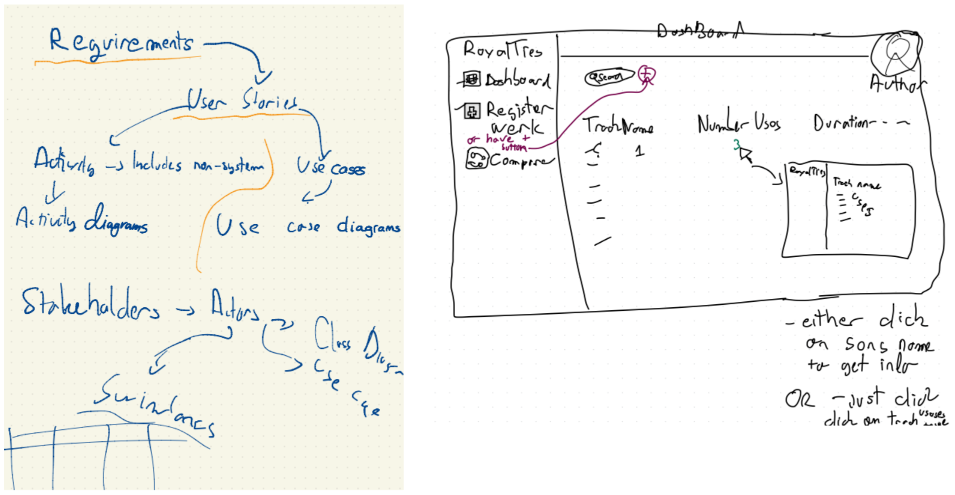


Figure 20 Wireframe diagram result via reflection process

Such reflection process was detailed separately. An internal meeting was organized, where team members would come up with a set of issues and solutions with regards to design. The result of such meetings was, each time, multiple design choices improved or implemented.

A subset of these questions was forwarded to the client during the requirement-gathering period. Unfortunately, many of these questions remained unanswered. In order to tackle this issue, the decision process meetings were raised in number, and solutions implemented for the issues we have encountered represent, at all times, our best possible approximation of the best-case scenario regarding each requirement that was deemed incomplete.

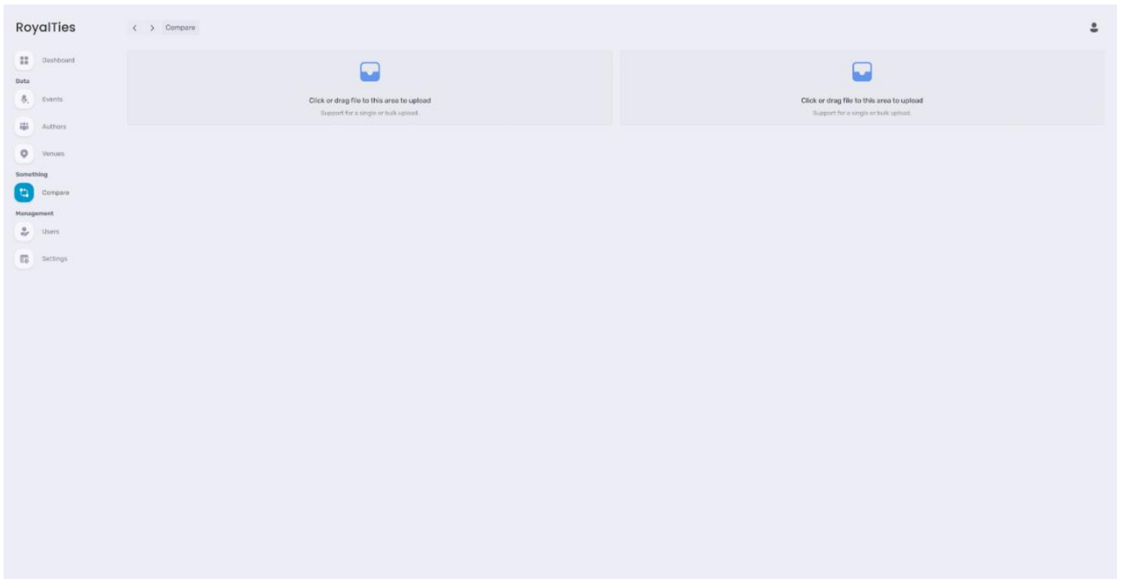

RoyalTies
Compare
Dashboard
Data
Events
Authors
Venues
Something
Compare
Management
Users
Settings
Click or drag file to this area to upload
Support for a single or bulk upload.
Click or drag file to this area to upload
Support for a single or bulk upload.

Figure 21 Prototype automated comparison upload page

6.2. Prototype reflection

The proposal constitutes the first prototype of the product. As mentioned, the reflection process brought about multiple changes and included new requirements that were not clear to us at the time of the initial development. Such changes will therefore contribute to the development of the product, during the following sprint, which resulted in our second Prototype.

The third prototype represents the final product, as delivered to the client at the end of the development phase. The design choices of such prototype are final, as presented throughout the document.

6.3. Initial vs Final design choices reflection

During our first MVP, we agreed to have multiple stakeholders interact with the dashboard. After investigating the accessibility considerations in this concern, we have decided to change the approach. It was particularly challenging to aggregate data about external authors, from the position of one author. We have decided to attach to each author the GEMA ID, and instead of adding their GEMA-related information manually (as depicted in the original dashboard from the client), we have decided to fill this data in the backend using a 'select' dropdown.

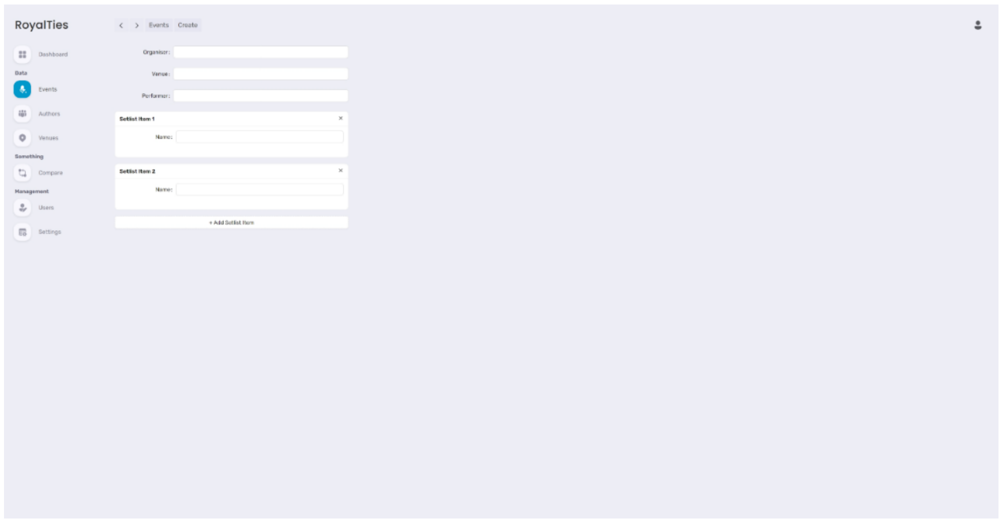


Figure 22 Prototype 2 - track list item UI

6.4. Reflection Meetings

Purpose

The aim of those meetings was to decide, after weeks of requirement elicitation, crucial design choices that were about to shape the course of the project. During the initial weeks, individual tasks were completed, and new design tasks were delegated such that in the end, a summary of weak, strong, and crucial points was formed, as well as the suitability of integrating such choices in the overall design and the effects that those actions would have.

It is worth mentioning that the requirements that were specified by the client were scarce, it was therefore commonly agreed that certain design choices were up to the developer, giving a commitment statement that such an effort would be our best possible solution to the said problem, given the circumstances. Starting as early as the first week of the project, the questions regarding the design choices that were discussed during the meeting were made clear to the client, yet unfortunately, some were left unanswered.

Crucial design decisions

1) Would the performer (the individual who performs the artistic act) be able to fill in the specific track, venue, and song-related data through our dashboard? Multiple viewpoints were presented by the members of the team. The initial standpoint was positive, yet multiple scenarios were later taken into consideration. An individual who does not have a management team, or is the organizer of the event, would still act as an Event Organizer. If the event is larger, a commitment statement is realized between the performer and the event organizer (Considering they are two different entities in this scenario). Given that this commitment statement can be a contractual, mutually agreed upon, it was decided to be drawn away, in order not to lose focus on the crucial requirements. In the absence of specific requirements from the client and taking into account the effects that this will have later on, it was decided that the performer should not be a separate user (with regards to authentication roles) in the application.

2) Following the aforementioned point, the Event Organizer will now be delegated the tasks and responsibilities of the entire organization of the event. Even though we tried to find downsides to this approach, it was commonly agreed that such points were futile, and the overall accessibility of the application would be improved if we followed this separation of concerns design choice.
3) Should the performer upload their own track-related data through the dashboard? – Given the multiple forms that GEMA has available for this aspect (original track registration, performance form, etc.). The decision-making process started with assessing the form fields from all of the GEMA forms. It is worth mentioning that, upon registration through GEMA, a performer gets a GEMA registration number. This aspect helped us abstract away all of the corner cases in which the individuals related to a song needed to be signed up to GEMA for track performance. Such data was decided to not be provided by the Event Organizer but added manually by the author of the song. To not clutter the database with relations, the featuring performers, composers and producers will be added via a dropdown, identifiable by their unique GEMA artist number (each of these artists would be, in technical terms, objects of class "Artist", differentiable by their attribute "Type" of type ENUM). The second most popular alternative was to aggregate the forms, where each individual performer would fill up their track-related data. Inspired by real royalties' distribution organizations, the sole performer that must be burdened with this task should be the original performer of the track, thus aggregating all said forms into one, with no bottlenecks, centralized information and no data leaks.
4) The users should not be able to edit their name, as every individual performer is linked to a unique GEMA account, and such action will create conflicts in the resulting documents, which could create issues regarding invoices.

After the decision-making part was concluded, tasks were delegated to the team members, each of whom had the responsibility of refactoring their previous work, let it be code, report entries or design diagrams. Another meeting followed the very next day, concluding the refactoring and evaluating the effect of the design choices.

7. Privacy & Security

7.1. GDPR

A subset of articles, relevant to our application yet not final and publishable, will be addressed within the privacy statement document, which is also available for the user to require informed consent for data access. It is important to note that such information is not legally evaluated for correctness, as well as not aligned with the current criteria of data manipulation consent of any data authority. The following policy statement is **by no means legally correct**, needs careful evaluation, and its scope is only for the reference of future developers.

"Start of privacy statement"

In order to comply to the most up-to-date regulations of data collection, GDPR and the officially requested items that need to be explained to the users [12], the scope of the following privacy statement is to showcase the way in which user's data is collected, manipulated or altered, and to inform all such users of their available actions that can be taken with respect to their collected information.

The representatives of the application RoyalTies, part of Kosmopolit Records, responsibility of Heger & Maurischat GbR.

The RoyalTies application will use the following sets of personal data from the users:

- **Login credentials**
  - In order to ensure the correct basic functionality of our application, the user is required to provide a valid email address and a password. The password is hashed using HMAC-SHA512, salted, and stored on the private database.
  - The other option is to login via Google OAuth2 standard, which does not require any login credentials.
  - The type of information provided is confidential and cannot be decrypted by any other regular users who interact with the application.
- **GEMA information**
  - The artist number and all the related information that is sent through, with respect to the artist number, will be saved such that correct interactions between users (aggregate, track comparison) can be achieved.
  - The type of information provided is external and can be attributed to one individual only.
- **Track list**
  - The tracks performed by the users will be saved on the backend side of the application for CSV comparison.
  - The type of information provided is performance related.
- **Venue location**
  - The locations of the performed tracks, which can subsequently be attached to an artist, will also be saved.
  - The type of information provided is geographically related.

Implications

- The data that the users will be asked for is the minimum amount of information needed for the correct completion of the tasks provided by the application, which the users will have to carefully assess and read upon registration.

- By registering with our application (by creating an account) the user agrees to this policy statement and has already carefully reviewed and agreed with the information provided.

Data processing

The data is securely stored in a private database, only available to the administrations of the application, and not visible to any other entity. The data is only processed internally, with the scope of ensuring a correct base functionality of the application and will not be shared with any other organization or 3rd party vendor, except for those whose integration is crucial for the correct behavior of the application (such as Stripe, where the emails are shared during checkout).

The users will have the following abilities and rights:

- Any user can require, via direct informed email, a complete deletion or hard copy of all records and entries of their data in the application.
- Disclaimer: The cascade deletion process of data has not yet been implemented. A user is able to delete their account, but their related data cannot yet be deleted.

Currently, the stored users' data will be kept as long as the account is valid. The only way to request a complete deletion of data is by direct informed email.

In the future, this will change to a fixed date, preferably a few months of inactivity.

"End of privacy statement"

7.2. Concerns

a. Data: The data that has to be manipulated via the application's interface is sensitive. Such data will have to be stored securely in the relational database and only manipulated via the cloud provider's official SDK.
b. Due to the multiple stakeholder groups, a concrete authorization mechanism will have to be set in place.
c. With respect to data manipulation, such actions must follow the most recent European directives with respect to data integrity. This will therefore require the interface to detail and mention the Privacy Policy throughout the web pages (preferably in the footer) and on the main sign-up/login page. When registering to the application dashboard, the user needs to agree to the privacy policy, only after they have thoroughly read it and agree to the points made in said document.

7.3. Privacy Policy

The privacy policy of the application will contain multiple headers, with each of whom the user needs to agree, in order to proceed with the sign-up process right on the initial page.

d. Entities: This section will enumerate the entities that take place in data manipulation, from the user to the developer and admin. Each of these entities will be given acronyms and will be referred to as such during the description of future sections.
e. Data storage: This section will explain how the user data is being stored, and which actions the user can take at any point in time, from adding, reviewing, modifying and deleting the data at any point in time.
f. GDPR: This section will expand upon the existing data directives available in the European Union and will describe each section of the GDPR with respect to the current application.
g. Usage to external parties: This section will describe how the data is being transferred over to external entities or third-party applications (e.g., Cloud Provider AWS or any other if applicable).

h. Miscellaneous stored data: This section will describe how additional data such as cookies or local storage elements are being used, and how they interfere with individual data.

8. Working prototypes

8.1. Prototype Week 3

During the initial MVP development phase, we were given initial requirements, yet not definitive. Even though the said MVP presented the basic functionality both us and the client anticipated, aspects such as security and UI/UX considerations were not present due to the design trajectory that we had in mind at that point in time.

The main purpose of this version of the final product was to outline the basic functionality for the most important features showcased by the MoSCoW methodology. As showcased in the resulting table (Requirement prioritization), we were first keen on contouring the authorization protocols for the basic stakeholders that were to interact with our application. During such process, aspects such as AWS implementation were unanimously decided upon to be left out of that specific print, and consequently out of this prototype version. Furthermore, it paved the way for how to tackle "could" requirements, such as the translation library.

Another aspect that was made clear throughout the development of such an MVP is the separation of concerns regarding the different stakeholders mentioned above. Different stakeholders resulted in the need for different user types, each with their unique actions.

One of the impediments present in this development process was the implication of CSV and GEMA form parsing, with respect to the user separation of concerns. Moreover, many details from such forms needed to be abstracted away, or aggregated, in order to facilitate an easy and coherent use for the users. Upon further investigation, it was decided that such a parsing mechanism was left out for the following version, to see which functionalities, attributes and methods each stakeholder was going to receive, such that the data that needed to be aggregated was made clear.

8.2. Prototype Week 5

The second prototype's development process revolved around the other aspects of the MoSCoW table that were not considered in the first one, namely implementing persistent actions, CRUD operations, authentication and authorization mechanisms.

Between the launch of the first prototype version and the implementation of the current one, the functional requirements were greatly altered, as more and more items appeared because of design reflection processes. Given that explicit requirements were lacking from the side of the client in this period, most of such requirements were reflected, altered, and decided upon internally. After the second version of the prototype was finished, the list of non-functional requirements was completed with items concerning page load, availability, and overall system performance indexes. This happened due to the team having a clearer overview of the separation of concerns and data manipulation, as a result of the reflection processes on the requirements that were unclear on the part of the client.

Some of the most highlighted deliverables during this process were the activity and use cases diagrams. A more in-depth explanation of the reflection upon such decisions is present in the Use-

Case Diagrams and Activity DiagramsThe resulting application, at the end of this development process, can be viewed at Appendix E: New UI

These were all results of the reflection process, both before and after the implementation of the MVP. The specifics, trajectory and design implications of such reflection process are described in the Reflection Meetings section.

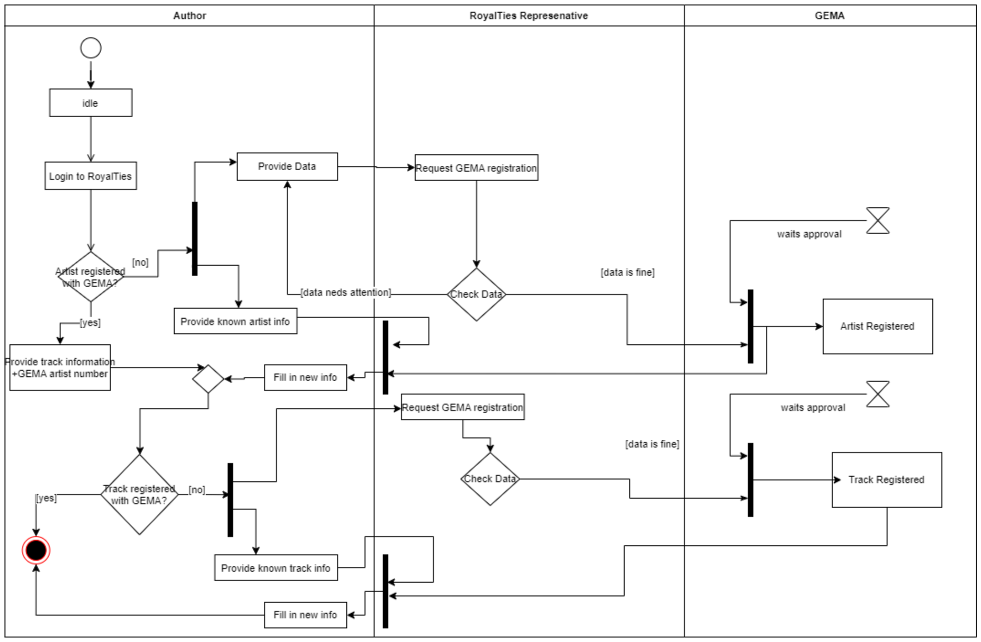


Figure 23 Activity Diagram showcasing the Author registration process.

At the end of the development phase of both prototypes, the client was informed about the process, and the subsequent status of the project, as well as relevant screenshots were made available for confirmation.

9. Evaluation

9.1. Metrics

UI/UX metrics

- The UI should be intuitive and should follow modern interface conventions.
- The overall website should follow a personality that is adequate for the end user (between playful and rigid).
- The UI components should follow a singular pattern and the navigation throughout the pages and components should be straightforward.
- The UX should facilitate the user of accessibility for individuals with auditory or visual impairments.
- The components should follow a primary/secondary UX convention for buttons and inputs, based on their target intended use within the component.

Architectural metrics

- The application should be accessible at any time.
- The application should scale as much as needed, and the load balancing should be done using modern conventions, and it shut down gracefully in any situation.
- The website's payment system should follow modern API security policies.
- The website should introduce authorization and should give the user the least amount of access possible to any functionality.
- The website's authentication system should be done via secure tokens and the credentials should be securely stored and encrypted, following the most up-to-date encryption family algorithms.
- The source code should be easily maintainable and developer friendly.
- The connection to the cloud provider should only be made through their Software Development Kit (SDK).
- The resulting architecture should be easily scalable, both horizontally and vertically.
- The requests should go through the reverse proxy, be verified using modern middleware techniques, and then rate-limited accordingly. Through this process, authorization and token validation actions should take place.

Frontend metrics

- The Author account user should not have access to the specific functionalities of Event Organizer account, and vice-versa. Intersected functionalities should still be accessible by both.
- The front-end architecture should be divided into easy to maintain components.
- The Balsamiq-resulted components should be compatible and easily maintainable with the front-end code.
- The front-end components should include all UI/UX characteristics mentioned in the section above.
- The frontend should only communicate with the backend through the secure API used to facilitate communication.
- The front-end interface should only improve the already existing design, while incorporating all the necessary features.

Backend & Database metrics

- The backend components should be accessible to the frontend components and should not respond to unknown requests.

- The backend components should only communicate with the cloud provider via up-to-date SDK features.
- Database entries should be encrypted and securely stored, following the privacy principles mentioned in that respective section.
- Database access should not be a bottleneck, and the connection should not be interrupted.
- Database access should only be provided to authorized personnel.
- No sensitive action should be accessible if the authentication middleware is not able to properly authenticate the request and its origin.

Security & Privacy metrics

- The user should be able to delete and modify their existing data at all times.
- The user should be able to view their data and be able to request a copy of their data if the function is not implemented.
- The user should acknowledge the privacy policy and ethical implications before signing up to use the application.
- User data should not be used in any situation except for the authorized cloud provider SDK methods.
- The manipulation of data should follow the most recent GDPR policies and restrictions.

Non-functional metrics

- The website should load in at most 2.5 seconds initially.
- The DOM structure should follow the best practices items, such as labelling each image with descriptive text., have non-hidden text etc.
- The web page should be resized appropriately, regardless of the zoom that the user inputs.
- Each page should load exactly once, and no other plugins or third-party processes should run alongside this load process.
- The database query for the entire index page of Author, Songs and Events should load in less than 3 seconds at any given time.
- The website should hold the user logged in indefinitely unless specified otherwise in the login page. Moreover, user session data should be deleted after they log out through the interface.
- The login phase should not take more than 3 seconds.
- The user should be able to switch between pages without noticeable delay.
- Moreover, the metrics mentioned in the section" Non-functional metrics" are also a source of truth for the testing process. To not reiterate them in this very section, we are going to consider all of them during testing. A subset of them have been verified during the Web App performance, SEO and accessibility testing, while others during the actual testing process (unit, system, etc.)

Miscellaneous metrics

- The testing plan will follow all the points in the sections. All of the Metrics will be evaluated separately, the process and the outcome will be documented in a separate table. The results will be evaluated, and in the case of noticeable difference, the design choices will be reviewed, and the application will be modified accordingly.
- Each section will be evaluated separately by everyone that was responsible for its implementation, and then the results will be commonly discussed to assess the process.
- The front-end and back-end evaluation will be done via an available developer platform and embedded libraries in .NET core.

- Integration testing will be done manually, documenting the multiple corner cases and their outcomes.

9.2. Testing

Methodologies

Unit testing

We used unit testing in order to check atomic functionalities and to verify individual components for correctness that could otherwise lead to unexpectedly hard to debug behavior. A subset of such unit tests is *CsvComparisonServiceTest,* ExcelExportServiceTest etc.

System testing

We used system testing in order to monitor the correct access and response time of the most crucial functionalities of our web application. This methodology is heavily inclined towards performance testing. An example of such an item is the Event Index Page - Loading Time, with both cache and no-cache accesses.

Integration testing

We used integration testing to ensure the correct workflow and relation between different components of our application. This is aimed towards user behavior, whose testing is achievable via Postman, API endpoints testing methodology.

Process

The main purpose of the aforementioned test is to preemptively catch any errors and unintended behavior regarding the web application flow. In order to create a **reflective** flow using these tests, the following methodology was set in place:

- The feature that is deemed important for testing is identified, subsequently followed by relevant metrics
- The test is implemented
- At the end of each sprint, the test is executed, and the results are assessed
    - If the result is a positive one, move to the next test
    - If the result is negative
        - Re-assess the test, to ensure it is up to date with the design choices
        - If affirmative, go back to the corresponding design choice of the corresponding feature, and reflect upon its design.
            - If the design is not considered appropriate (anymore), reflect upon the issues that led to this problem to arise in the first place.
            - Reflect on the new design choice, re-design the feature and make the appropriate changes to the test.
            - Re-run all other tests
        - If negative, reflect upon the new implementation given the specific requirements and design choices.
- As this is part of the reflection process, the changes were not explicitly documented separately, but included in the Reflection Meetings and Prototypes sections, based on the time the issues were fixed.

Results

The following test suites were composed, given the aforementioned metrics. The suits focus on the most crucial aspects of our application, which should follow the correct behavior at any point in time.

Table 3 Testing results table

Type	Name	Metric(s)	Expected outcome	Actual Outcome
Unit	CsvComparisonServiceTest	Required functionality	All tests passed	All tests passed
Unit	ExcelExportServiceTest	Required functionality	All tests passed	All tests passed
System	Event Index Page - Loading time (No Cache)	Performance	800 milliseconds	611.64 milliseconds
System	Event Index Page - Loading Time	Performance	400 milliseconds	464.62 milliseconds
System	GDPR User Deletion Test	The user should be able to delete and modify their existing data at all times.	Successful deletion	Successful deletion
Integration	OrderableServiceTest	User should be able to reorder tracklist elements?	All tests passed	All tests passed
Integration	Postman API test	The GraphQL query for the entire index page of the main dashboard page should load in less than 3 seconds.	200ms	36ms

9.3. Web Performance. SEO, Accessibility & PWA audit

For Web Performance, we have chosen the following metrics to assert the optimization index: page load time, responsiveness, resizing (as mentioned in the non-functional requirements description), and the metrics for PWA (Progressive Web Application) provided by Google Lighthouse extension [13], embedded within chromium browsers.

It is important to note that the metrics were not consulted before the testing phase, and the metrics that were and will be presented throughout this document have specifically resulted from design reflection processes.

These metrics were chosen such that the application would be tested thoroughly in the development phase, given the lack of customer information, user data and traffic audits.

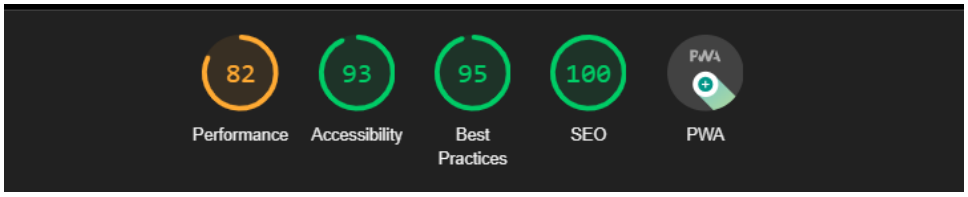
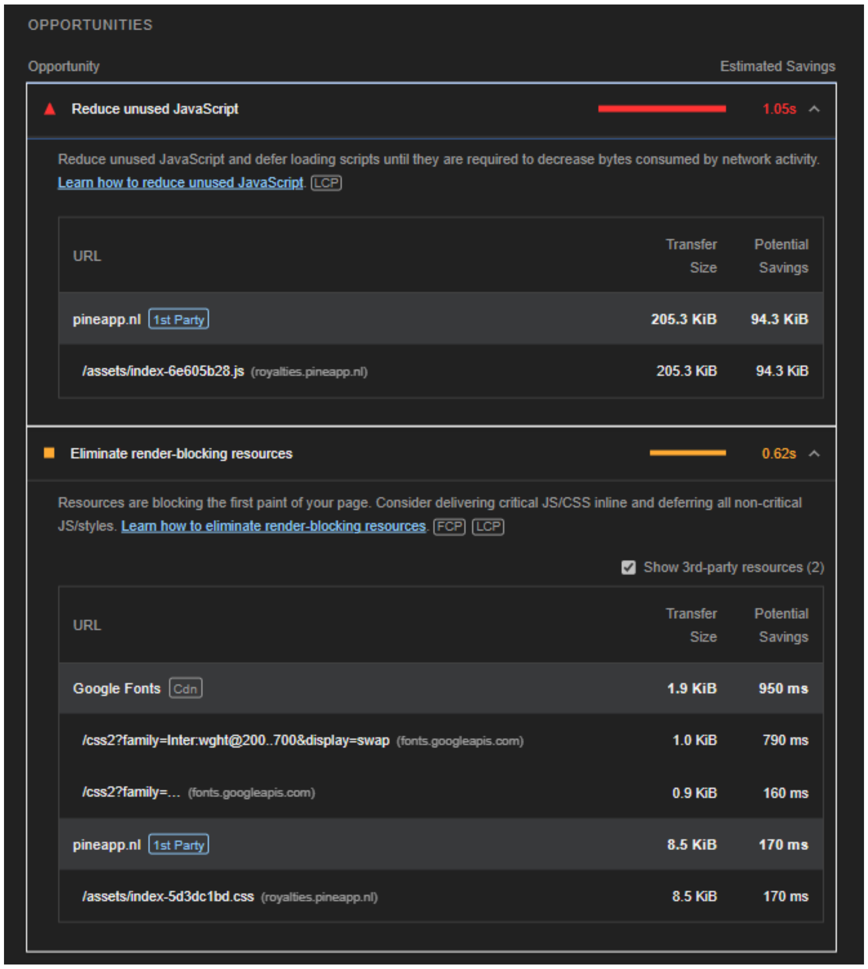


Figure 24 Overall audit of the website, per Google Lighthouse embedded extension

Performance evaluation

With regards to Web Performance, the principal issues were inefficient asset bundling, "render-blocking resources" and inefficient caching layers. Such issues were not part of the non-functional requirements item list, as it is only feasible to test post-deployment. Given this reason, such tasks were not focused on, and instead the development process (code-wise) was more focused on "Accessibility" and "Best Practices".



OPPORTUNITIES

Opportunity — Estimated Savings

Reduce unused JavaScript — 1.05s

Reduce unused JavaScript and defer loading scripts until they are required to decrease bytes consumed by network activity. Learn how to reduce unused JavaScript. LCP

URL	Transfer Size	Potential Savings
pineapp.nl 1st Party	205.3 KiB	94.3 KiB
/assets/index-6e605b28.js (royalties.pineapp.nl)	205.3 KiB	94.3 KiB

Eliminate render-blocking resources — 0.62s

Resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles. Learn how to eliminate render-blocking resources. FCP LCP

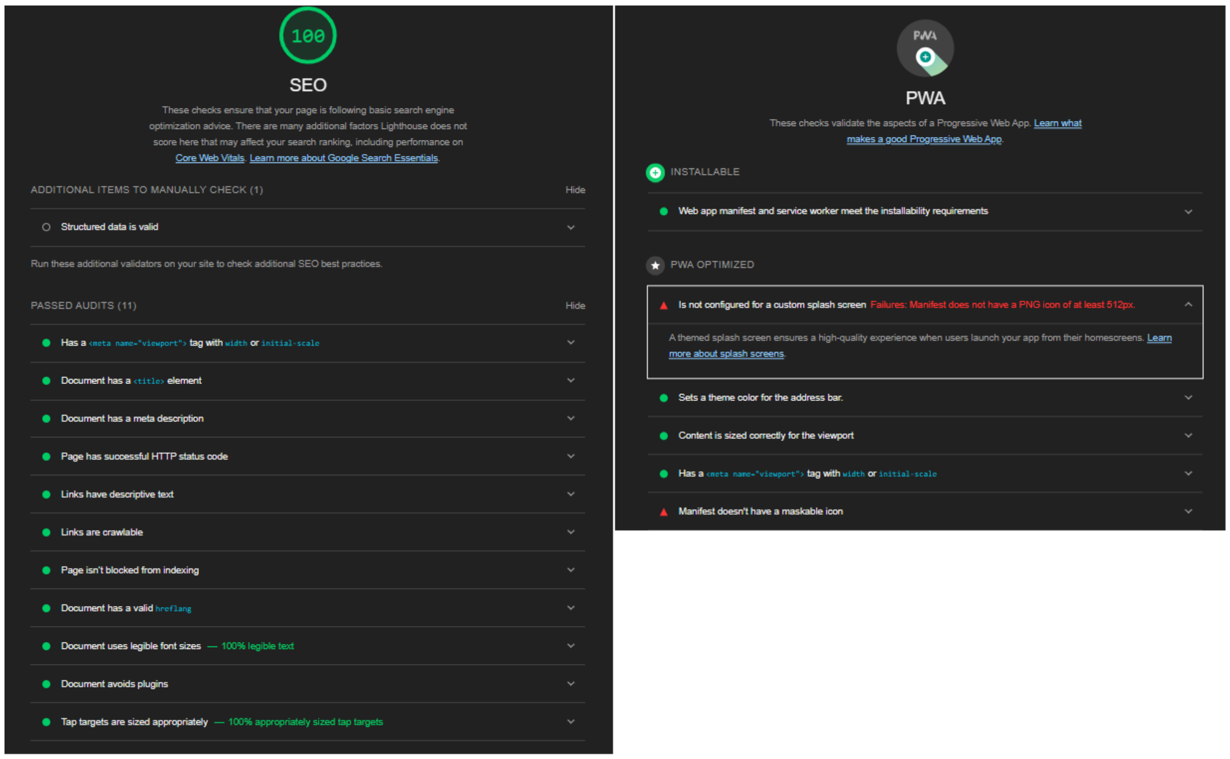
Show 3rd-party resources (2)

URL	Transfer Size	Potential Savings
Google Fonts Cdn	1.9 KiB	950 ms
/css2?family=Inter:wght@200..700&display=swap (fonts.googleapis.com)	1.0 KiB	790 ms
/css2?family=... (fonts.googleapis.com)	0.9 KiB	160 ms
pineapp.nl 1st Party	8.5 KiB	170 ms
/assets/index-5d3dc1bd.css (royalties.pineapp.nl)	8.5 KiB	170 ms

Figure 25 Performance metric weak points

SEO & PWA metrics

General best practices such as assigning a label to an image, making a custom URL or route crawlable, page load time, readability, and page resizing were taken into consideration, as mentioned in the non-functional requirements section. The images presented in this section only showcase the imperfections regarding such metrics, which pose a testimonial to the correct implementation of the mentioned requirements (except for the ones that will need further testing post-deployment, such as traffic handling).



100

SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.

ADDITIONAL ITEMS TO MANUALLY CHECK (1) Hide

○ Structured data is valid

Run these additional validators on your site to check additional SEO best practices.

PASSED AUDITS (11) Hide

- ● Has a `<meta name="viewport">` tag with `width` or `initial-scale`
- ● Document has a `<title>` element
- ● Document has a meta description
- ● Page has successful HTTP status code
- ● Links have descriptive text
- ● Links are crawlable
- ● Page isn't blocked from indexing
- ● Document has a valid `hreflang`
- ● Document uses legible font sizes — 100% legible text
- ● Document avoids plugins
- ● Tap targets are sized appropriately — 100% appropriately sized tap targets

PWA

These checks validate the aspects of a Progressive Web App. Learn what makes a good Progressive Web App.

INSTALLABLE

- ● Web app manifest and service worker meet the installability requirements

PWA OPTIMIZED

- ▲ Is not configured for a custom splash screen Failures: Manifest does not have a PNG icon of at least 512px.

  A themed splash screen ensures a high-quality experience when users launch your app from their homescreens. Learn more about splash screens.
- ● Sets a theme color for the address bar.
- ● Content is sized correctly for the viewport
- ● Has a `<meta name="viewport">` tag with `width` or `initial-scale`
- ▲ Manifest doesn't have a maskable icon

Figure 26 SEO and PWA evaluation

10. Risk analysis and Contingency planning

1. Data Security and Privacy Violations:

- Details: With sensitive user data, there is a risk of unauthorized access, data breaches, or non-compliance with privacy laws.
- Mitigation:
    - Implement end-to-end encryption for data in transit and at rest.
    - Regularly update and patch system vulnerabilities.
    - Conduct periodic security audits and assessments.
    - Ensure compliance with GDPR and other relevant privacy laws.
- Contingency:
    - Have an incident response plan ready.
    - Notify affected users and authorities in case of a data breach.
- Investigate and solve the security issue promptly.

2. System Downtime and Unavailability:

- Details: System downtime can lead to loss of user trust and financial losses.
- Mitigation:
    - Implement system redundancy and failover mechanisms.
    - Use reliable cloud services with high availability guarantees.
    - Conduct regular system maintenance and monitoring.
- Contingency:
    - Communicate transparently with users during downtime.
    - Have a disaster recovery plan to restore services quickly.

3. Inaccurate or Incomplete Data Input:

- Details: Inaccurate data can lead to incorrect royalty calculations and payments.
- Mitigation:
    - Develop input validation mechanisms to ensure data accuracy.
    - Provide user guidance and support for data entry.
    - Implement data review and correction processes.
- Contingency:
    - Allow for data corrections and adjustments post-submission.
- Implement a dispute resolution mechanism for data and payment disputes.

4. Ineffective Multilingual Support:

- Details: Language errors or lack of support can alienate non-English speaking users.
- Mitigation:
    - Test and verify language support thoroughly.
    - Implement a process for users to report language issues.
- Contingency:
    - Quickly correct reported language errors.
    - Consider expanding language support based on user needs and feedback.

5. Non-compliance with GEMA and Other Regulatory Requirements:

- Details: Non-compliance can result in legal penalties and loss of user trust.
- Mitigation:
  - Stay updated on GEMA and other regulatory requirements.
  - Implement compliance checks and documentation processes.
- Contingency:
  - Work with legal counsel to address compliance issues promptly.
  - Make necessary system adjustments to ensure compliance.

6. Project Overruns Budget

- Details: Unanticipated costs and mismanagement of funds can cause some budget issues.
- Mitigation:
  - Establish a well-structured budget plan and implement cost-control measures.
- Contingency Plan:
  - Maintain a financial reserve to cover unexpected costs. In the worst-case scenario, adjust the project scope.

7. Lack of Stakeholder Engagement

- Details: Insufficient involvement from key stakeholders can lead to a lack of direction and approval for the project.
- Mitigation:
  - Identify and engage crucial stakeholders early in the project and try to address their concerns and expectations proactively.
- Contingency Plan:
  - Implement strategies to re-engage disinterested stakeholders and possibly bring in new stakeholders to support the project.

8. Communication Breakdown

- Details: Poor communication within the team or with stakeholders can lead to misunderstandings which can potentially cause significant problems.
- Mitigation:
  - Establish effective communication channels and protocols.
- Contingency Plan:
  - Designate communication coordinators to facilitate information flow.

9. User Resistance to the new system

- Details: End-users may be resistant to adopting the new system due to lack of understanding or fear of change.
- Mitigation:
  - Engage end-users early in the development process.
  - Provide training and support.

- Contingency Plan:
    - Develop and implement a change management strategy to facilitate user adoption and acceptance of the new system.

11. Commitment statement

It was mentioned to the client that the submission and the agreement of such submission, therefore, will constitute our statement of commitment to respect the requirements and deliver our best work during the development phase of the product. This commitment statement is only valid throughout the course of the development process (until November 10th, 2023). **None** of the students assigned to this project can be held accountable for any future issues concerning the implementation, performance, and design of this product.

No remarks regarding this statement, at any point in time until November 10th, 2023, will be considered an agreement by all parties. The commitment statement can be subject to change throughout the 10 weeks of development, yet regardless of its status, no student can be held accountable.

12. Manual

12.1. Login/Register

When first visiting the webpage of the product, users are prompted with a login page. This includes a forward link to create a new account given that the current user has not created one. Moreover, a user has the ability to request a new password if they had forgotten it. This page also contains the policy statement, to ensure that users grasp all the required information about the privacy implications of the website they are going to browse. It is also possible to log in through Google OAuth2. The register page contains a form with user and password fields, as well as the possibility to sign up through Google OAuth2, for a more secure alternative. Please refer to the Frontend section of this document, for a more detailed description of the pages.

12.2. User scopes & Actions

The authorization mechanism for the website revolves around the type of users, and their respective actions that they can follow, given their role in the GEMA environment. This differentiation comes from the responsibility distribution of artists, event organizers and all other entities involved in the lifecycle of a track list.

The list of available user scopes, as well as their available actions are listed below. The respective pages that contain such actions are described later on in this section.

12.3. Actors

Artist

- List all tracks: A user can see an indexed list of all the tracks that he is registered for, as well as sort such list.
- Create, edit, view, delete track: To comply with the GDPR and CRUD baseline operations, and user is, at all points, able to add, view, edit or delete an item in the indexed list of tracks. Please refer to the GDPR section for a detailed description of what privacy items this section includes. The relevant UI and UX elements that are comprised within these actions are described in the Frontend section of Application Design.
- Pay for subscription: The user can pay for a subscription, via Stripe. The features that distinguish between a base and paid subscription account are up for consideration. Please refer to the Application Design section for a detailed description of the payment process, and the underlying features.
- Change their email address + password: Through the settings webpage of the dashboard, one can modify their email address and the password.
- Multi-factor authentication (2FA): An artist is also able to sign in via multi-factor authentication, for a more secure mechanism.

Organizer

- Compare track lists: As opposed to a normal user, an event organizer can compare the track list for an artist, see the differences and similarities, which can invariably help in determining the payment gaps and keep track of royalty's distribution differences. Such action can be done via both upload or drag-and-drop actions, require two CSV files and exports a Gema complaint form Excel file, in case there are any missing performances.

- List events: An event organizer can see, view, delete and add a list of live events, each corresponding to an artist. Please refer to the Application Design sections in order to understand the workflow, and the Frontend section to see the relevant UI elements.
- View/delete/add base operations for tracks: Once again, in order to respect the GDPR perquisites with respect to data manipulation, an event organizer will be able to view, delete, and add a track. Please do refer to the Frontend section for a more detailed guide regarding the corresponding UI components.
- Select existing track list within an event, ordering, list existing tracks: such operations are only available from a bird-eye view by the event organizer, and not accessible to the regular author account. Tracks are indexed and sorted, and the EO is provided an additional interface in order to manipulate this data. The UI components for such actions are also described in the relevant Frontend section, extending the basic view UI components of the author.
- Overview list of venues (global): Based on the requirements and features available to each agent that interacts with the application, it was decided that an author should not be burdened with the aspect of managing venue entries. Therefore, the event organizer is also responsible for such tasks, as well as being allowed to view, edit or delete any venue.
- See billing statuses: As the event organizer is the main stakeholder whose job is to compare the CSV files, and to notice any missing items, they can also check the billing statuses of such items. The functionality of CSV form comparison is extended with this dashboard of tracking the status of royalties-based billings, which is further explained in the Frontend section.
- Pay for subscription & Change email address + password + 2FA: These actions completely resemble the functionalities available to the base user, an author, and are not extended with any additional information, features or UI components.

Admin

- Overall functionality access: The admin role benefits from all the features mentioned above, extending all functionalities presented for authors and event organizers. All the UI components are available to admins, which can delete any information that is not directly related to the account of the author or event organizer. The admin view tabs in the Frontend section showcase these functionalities in detail.
- Indexed lists of artists and active users: Additionally, to the global functionality access, the admin account is also able to an indexed (or ordered) list of active users, authors or event organizers. They also have the ability to manipulate the access of each user within the scope of the application.
- Settings page: Available only to admins, the settings page provides a set of items that reside outside the visibility scope of the event organizer, and which relate to organizational-related elements, within a separate dashboard page.

12.4. Dashboard Pages

The following pages are available through the dashboard: Login/Register page, Dashboard page, Events page, Artist's page, Venues page, Compare tool page, Users page and Settings page. Each of these pages is detailed individually in the Page description subsection of Application Design/Front-end. Please refer to each individual item for more relevant information.

12.5. Interface accessibility

- Language: Each user, regardless of their type of account, can select between English, Dutch, and German language modes, which will change the translate all UI components and text automatically.
- Day/Night mode: The users can switch between day and night mode interface themes, which will change the contrast and color palette of each individual UI component and text box throughout each dashboard page.
- Download data: The users can download their CSV related data, such as track lists and missing payment
- Policy page: All of the users will be prompted with a reference to an external page on the register/login page, relating to the policy items, privacy and security agreements and all usage of data that

13. Reflection considerations

Although the product is deemed satisfactory by the team members, there undoubtedly is a set of actions and decisions that, after careful consideration at the end of the development phase, have been addressed. Should we be given the chance to do it again, we would change the following:

- The communication medium: It was deemed crucial, throughout the development phase, that a clear understanding of requirements should be the main focus. As we were forced to design a subset of such items, we would be focused on enforcing the communication mediums, protocols and schedule from the very beginning.
- Design choices per actor, and not general design choices: As mentioned throughout the decision making and thought process sections, some design choices were greatly altered by the actor usage. Even though we took the correct approach in designing such items, we have realized too late the importance of stakeholders that are going to interact with the application. Such action had the consequence of us needing to redesign the priority of design choices, in order to integrate such clear requirements.
- The use of subtasks: Even though time management was not a problem throughout the development phase, the usage of subtasks was proven efficient too late in such phase. We have realized that taking hold of a big task would provide a greater psychological burden, and therefore the solution to such a task would be either unclear or taken too long to complete.
- Client Engagement: The level of engagement with the client could have been more consistent. Regular check-ins and update sessions would have ensured that the development was on the right track based on the client's expectations. Moreover, it is important to note that this limitation was caused by the lack of communication on the client side, which was explained extensively at the beginning of Section 3.
- Quality Assurance (QA) Process: The establishment of a comprehensive Quality Assurance process from Sprint 1 would have ensured a higher standard of quality for the deliverables. Incorporating systematic testing, code reviews, and other QA practices throughout the sprints could have potentially identified and rectified defects early on.
- Accessibility: By incorporating accessibility standards and guidelines into the design process from the beginning, the final product could have a higher accessibility standard so that it can be used by a range of people including those with disabilities.

14. Future development and maintenance

The application described is the result of a limited working period. Throughout the sprints mentioned in the planning section, the work has been extensively brushed upon and changed according to requirement elicitation processes. The channel for gathering such requirements was uncertain, and the initial set of requirements had arrived rather late in the design process. The resulting requirements were the best possible estimation for the individuals that have been responsible for gathering them.

Moreover, the future development and maintenance of this project was ensured by following the maintainability design principle throughout all sections.

To aid in the further development of the application, some design choices such as the architectural structure also provide an extension for further design. One prominent example in this regard is the use of AWS services, as described in its respective section. However, alongside the client, it has been agreed that the deployment of this application on AWS cloud provider was unfeasible in the narrow period for this project. To facilitate the future development of the application following this route, the following AWS section describes the further steps that need to be taken to achieve such a deployment phase and maintain it even further.

14.1. AWS deployment

As Amazon Web Services provide the best current approach to web app deployment, we advise that this product should make use of its features. The underlying implementations should definitely be revised by experienced senior developers. AWS can be used as a cloud provider, as it's the current base state of the art choice for such applications. The data has to be properly secured, as it is of great sensitivity. RDS can achieve this using its relational style and security. As uploaded data can be very large, the throughput can grow exponentially and therefore the load balancers and elastic containers provided by AWS will be able to match the requirements. The alternatives would be Azure, which has smaller market coverage, and yet could make the deployment process for further development less maintainable and provide a steeper learning curve for future developers.

Performance improvements

The section Web Performance. SEO, Accessibility & PWA audit presents a systematic overview of the items that can be improved, regarding performance. To re-iterate these items, possible performance improvements can be:

- Analyzing a better bundling option
- Pre-loading CSS assets
- Performing a CDN check and edge caching the assets mentioned in the audit
- Add remaining optimization files such as a manifest JSON.
- Double-check JS behavior.
- Ensure a secure database communication implementation and protocol usage
- Edit and correct the GDPR and privacy policies to use official terms and statements
- Revisit the architecture, perhaps hire a consultancy firm to evaluate the appropriateness
- Ensure to fix the remaining performance steps to incorporate appropriate PWA structure

14.2. Testing

As the application makes use of sensitive and highly personal data, it is recommended that further testing should be performed to evaluate the state of the product. Current tests should be evaluated for coherency with respect to the business goals, as well as revised within the codebase.

Moreover, metrics should be evaluated in order to make sure they are aligned with the end goals of the company. The metrics mentioned in the "Metrics & System testing" were evaluated during the final evaluation step, yet the degree of appropriateness should also be evaluated from the perspective of the client.

15. Individual Contribution

The following table shows the individual task distribution, corresponding to each team member. Meetings were organized to compile a list of tasks, in bulk. These tasks were then delegated to Trello, in multiple swim lanes and were assigned to each person individually. Each individual task was performed during the specific sprint it is part of. The order does not reflect the actual implementation prioritization.

The following table items were compiled given the strengths and weaknesses of each team member, which was the subject of discussion during the first week of development.

Table 4 Individual Contribution

	Section/Subsection
David Galati	- Initial Trello Board / Assigning Tasks - Introduction, Problem Description - Stakeholder identification - Requirements elicitation - User Stories Derived from Requirements - Preliminary Class Diagrams - Use Case Diagrams and their descriptions. - Activity Diagrams and their descriptions. - Poster Design -
Michiel van Huijstee	- Development of frontend - Development of backend - Database design - Frontend design - Software manual - Development and execution of tests - Security & Privacy considerations - UI/UX user considerations - Deployment process
Sebastian Mihai	- Tools (database, backend, UI/IX, versioning, task division, hosting, testing, payment) - Security & Privacy considerations - Metrics - GDPR (privacy statement) document - Market Insights

	- Historical data - Requirement specification - communication issues section - (multiple sections) GEMA submission procedure - Future development and maintenance + AWS considerations - Web Performance, SEO and accessibility metrics using Lighthouse - Testing plan + scenarios + benchmarks - Architectural design – Database (only design section) - UI report section - Accessibility & UX + considerations - Design implications, Individual implications - Frontend report section (not the code), TSX components section - Current market solutions, competitors, current insights - Domain design section - Proposal and Prototype reflection - Initial vs Final design choices - Initial task division and project structure (meanwhile reformatted) - Reflection on Design Choices - Tools, models, choices, 3rd party integration justification. - Ethics, terms & conditions - Software Manual - Prototypes reflection + description - Description of frontend pages - Justification of overall design choices - Coordination with the client - Deployment section (not actual deployment process) - Planning - communication medium - Design decisions reflection meetings section - Redesign and link together different technical sections of the report (metrics, requirements, testing) - Resolving comments, fixing other sections, formatting the document - Description of old system + GEMA submission, complaint and results CSV form description - Conclusion section - Commitment statement - Referencing, table/images reformat - Glossary of terms + design section terms
Brianna Drîngă	- Project planning, setting up milestones and deliverables, project scope - Agile development methodology + justification of choices - Gnatt Chart - Wireframes using Balsamiq - Security by Design consideration – SBD checklist - Redesigning report structure - Final presentation - Final report changes on multiple sections (e.g. Future development, Tools and Models etc.)
Radmehr Ghadiri	- Risk Analysis and Contingency planning - Product Description - Requirements - Interview Report - Requirement and Use Case Table - Redesigning report structure - Gnatt Chart - Moscow structure for requirements - Wireframes using Balsamiq

	- Reflection considerations - Requirements -> Use Cases - Conclusion - Final minor report changes on multiple sections

16. Conclusion

This document has highlighted in detail the development process of designing the product for the company RoyalTies. The end product is a functional web application, through which different stakeholders in the music industries can automate the process of handling, comparing, and assessing the royalties' distribution of their songs. The final outcome is an improvement of the pre-existing system and adds new features meant to ease the lives of performers, publishers, and event organizers. The core business idea of this Software-as-a-Service web application is the automated processes that could allow performers to become independent entities with regard to copyright and royalties' distribution processes. In the end, by following clear monetizing criteria, which are to be determined by the client and are outside the development team's scope, we can meet end goals and business criteria that would allow further development of the application. In summary, this report thoroughly documents the careful design process that has led to the delivery of RoyalTies' web application, an essential tool to streamline the royalty management workflow for its users.

17. References

[1] "Welcome to ASCAP - the world leader in performance royalties, advocacy and service for songwriters, composers and music publishers," www.ascap.com. Accessed: Nov. 04, 2023. [Online]. Available: http://www.ascap.com/

[2] "Broadcast Music, Inc. (BMI)," BMI.com. Accessed: Nov. 04, 2023. [Online]. Available: https://www.bmi.com/

[3] "REWNu | DE-Parcon." Accessed: Nov. 02, 2023. [Online]. Available: https://www.de-parcon.de/de/rewnu.html

[4] "What is a setlist?," gema.de. Accessed: Nov. 02, 2023. [Online]. Available: https://www.gema.de/en/help/users/use-music/setlist/what-is-a-setlist

[5] "Eine Setlist bei der GEMA einreichen," gema.de. Accessed: Nov. 02, 2023. [Online]. Available: https://www.gema.de/de/hilfe/musiknutzer/musik-nutzen/setlist/wie-reiche-ich-eine-setlist-ein-

[6] "Setlist einreichen bei der GEMA: So geht's," gema.de. Accessed: Nov. 02, 2023. [Online]. Available: https://www.gema.de/de/musikurheber/online-services/meine-setlists

[7] "Claim," gema.de. Accessed: Nov. 02, 2023. [Online]. Available: https://www.gema.de/en/music-creators/royalties/claim

[8] "GEMA Onlineportal." Accessed: Nov. 02, 2023. [Online]. Available: https://www.gema.de/portal/app/dashboard

[9] "gema-satzung-nach-mitgliederversammlung-2023_en.pdf." Accessed: Nov. 02, 2023. [Online]. Available: https://www.gema.de/documents/d/guest/gema-satzung-nach-mitgliederversammlung-2023_en

[10] "So funktioniert der GEMA Soundfile Upload," gema.de. Accessed: Nov. 02, 2023. [Online]. Available: https://www.gema.de/de/musikurheber/repertoire/werke/soundfile-upload

[11] ByteHide, "Implementing Clean Architecture in .NET Core," ByteHide Blog. Accessed: Nov. 02, 2023. [Online]. Available: https://www.bytehide.com/blog/clean-architecture-net-core

[12] "What information must be given to individuals whose data is collected?" Accessed: Nov. 02, 2023. [Online]. Available: https://commission.europa.eu/law/law-topic/data-protection/reform/rules-business-and-organisations/principles-gdpr/what-information-must-be-given-individuals-whose-data-collected_en

[13] "Lighthouse overview," Chrome for Developers. Accessed: Nov. 02, 2023. [Online]. Available: https://developer.chrome.com/docs/lighthouse/overview/

18. Appendix

18.1. Glossary of Terms

1. ASCAP: The United States based society responsible for rights distribution (Welcome to ASCAP - the world leader in performance royalties, advocacy and service for songwriters, composers and music publishers)
2. BMI: Large United States based organization involving interpretation rights (BMI | BMI.com)
3. AWS: Amazon Web Services, a popular cloud-based platform, with state-of-the-art architectural and deployment features (Cloud Computing Services - Amazon Web Services (AWS))
4. MVC: Model View Controller architectural pattern
5. Django: Popular MVC Python-based framework (The web framework for perfectionists with deadlines | Django (djangoproject.com))
6. API: Application Programming Interface, used for data transfer
7. SDK: Software Development Kit
8. CSV: “Comma-separated values” data document format
9. SaaS: Software as a Service
10. MVP: Minimum Viable Product (minimalistic version of a working product)
11. WIP: Work in Progress
12. Cloud-based: When a software application is stored remotely on a hard drive, in a physical location and managed by an external provider, it is often referred to as “cloud-based” application.
13. Stripe: One of the most popular payment services (Stripe | Payment Processing Platform for the Internet)
14. Architecture: In the context of this product, it is referred to as the backbone of the application
15. UI/UX: User Interaction/ User Experience. UI refers to elements regarding direct interaction between frontend and the user, while UX refers to the underlying process.

18.2. Appendix A: Interview Reports

Meeting 1

Meeting Summary

The meeting held on Friday, September 8, 2023, focused on discussing the key aspects of the GEMA project, a royalties management system for music authors and artists in Germany. The primary objective of this meeting was to outline the main functionalities and requirements of the project, determine the target users, consider the technology stack, and address some important considerations for the project's success.

Product Overview

GEMA: GEMA is a royalties company based in Germany that manages and distributes royalties to music authors and performers. It provides a statement that details when and where a song has been used, including live events, radio distribution, and other instances of intellectual property utilization.

Royalties Distribution
It's the promoter of live events who is responsible for paying royalties, not the performers themselves. A portion of the income from ticket sales is redirected to cover these royalties. It also should be noted that the system is not representing the music authors when the song is live, but also radio distribution and all places where your intellectual property is used. In the United States, it is possible to sell the entire rights to a song, but in Germany (DE), performers always retain the copyrights. To manage these rights, an association like GEMA is needed.

Software Functionality
The primary goal is to create a platform that allows music authors and performers to manage their royalties more effectively. Key functionalities include:

Publishing Company Control: The software should cater to publishing companies that operate internationally, with various labels. It should provide a comprehensive control system for managing royalties distributed for events each year.

Individual Artist Management: The software should also cater to individual performers who do not work with a company. It should enable them to easily manage their song lists and compare them with GEMA's distribution chart (CSV) to identify missing data.

Data Entry and Schema: Authors will fill in data within the app, using a structure similar to GEMA's schema. There will be no need to upload data separately.

User Base: The target user groups include music publishers, artist management, and music authors. Both individual artists and music publishers can create accounts.

Technology Stack
The choice of technology stack is flexible and can be determined based on project requirements and constraints

Data Validation and Automation
To ensure data accuracy, the system will require users to list every concert they play in the GEMA online system. On January 1st of each year, GEMA will provide a CSV file with payment data. This data will be compared with the user's own system to identify missing dates. The system will help automate the process of identifying missing information and discrepancies between the user's data and GEMA's data.

Additional Considerations
Cost Efficiency: AWS was considered too costly for the project. An architectural overview and alternative solutions will be provided.

Contract: The need for a contract for source code ownership and related issues will be discussed with the supervisor.

Subscription Management: The system will incorporate subscription management and accounting, potentially involving artists' payments.

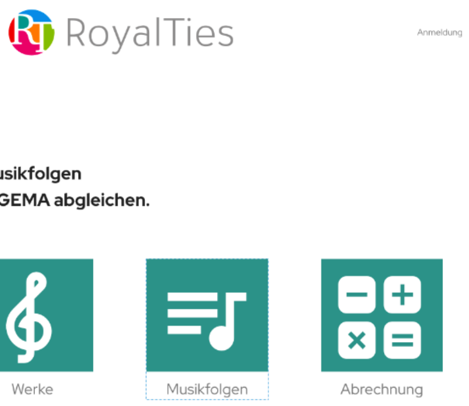
User Dashboard: The user dashboard will encompass all discussed functionalities, including password management.

Payment Management: The system will facilitate payment management and highlight login issues or other problems.

Multilingual Support: The project aims to offer services in both German and English, necessitating the inclusion of a library or framework for translation.

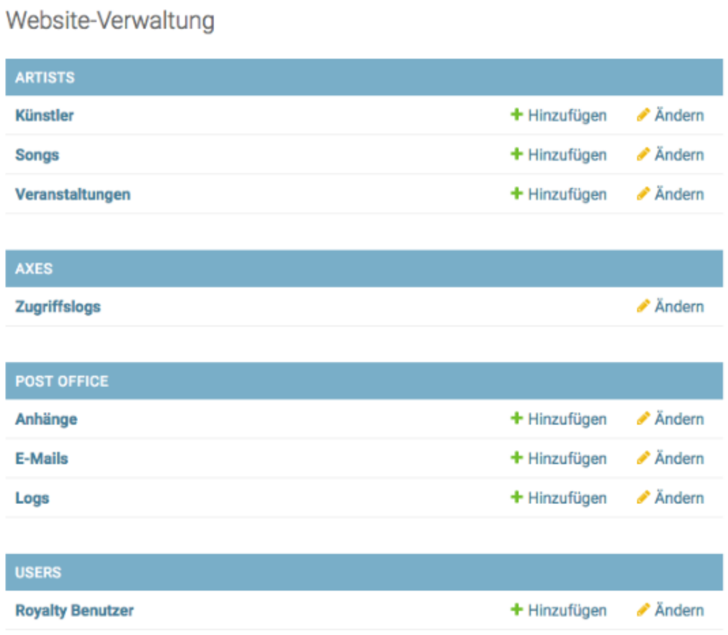
Conclusion
The meeting ended with a commitment to proceed with the project, considering the outlined functionalities, user requirements, technology stack, data validation, and additional considerations. The goal is to create a user-friendly and efficient platform that simplifies the management of royalties for music authors and performers in Germany. The importance of making the system engaging and enjoyable for users was emphasized to ensure its success. Further discussions and planning will continue to shape the project's development.

18.3. Appendix B: Initial RoyalTies



Wilkommen auf RoyalTies!
Bei uns kannst Du deine Musikfolgen
einfach und sicher mit der GEMA abgleichen.

Figure 27 C1: Main page of the application


Website-Verwaltung

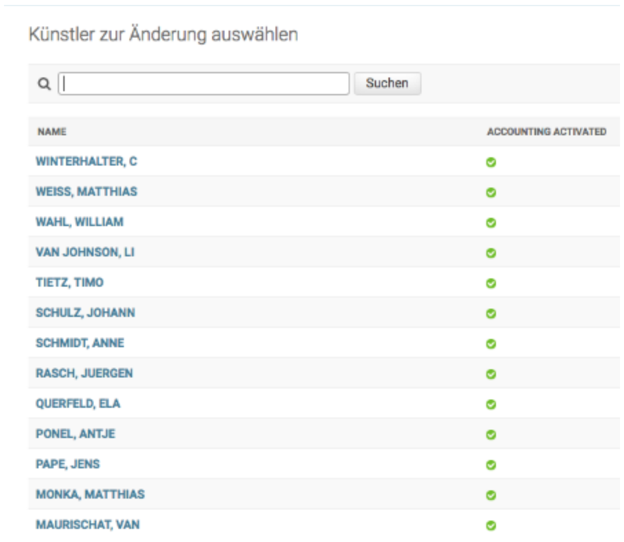
ARTISTS		
Künstler	+ Hinzufügen	Ändern
Songs	+ Hinzufügen	Ändern
Veranstaltungen	+ Hinzufügen	Ändern

AXES		
Zugriffslogs		Ändern

POST OFFICE		
Anhänge	+ Hinzufügen	Ändern
E-Mails	+ Hinzufügen	Ändern
Logs	+ Hinzufügen	Ändern

USERS		
Royalty Benutzer	+ Hinzufügen	Ändern

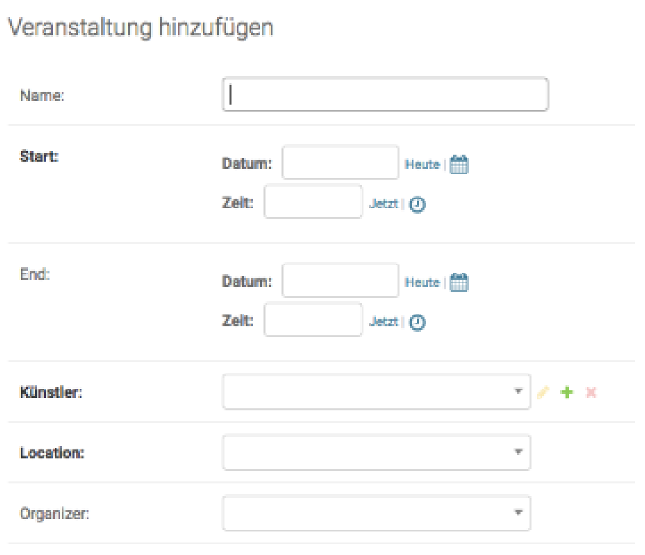
Figure 28 C2.1: Old administration Django component


Künstler zur Änderung auswählen

Suchen

NAME	ACCOUNTING ACTIVATED
WINTERHALTER, C	✓
WEISS, MATTHIAS	✓
WAHL, WILLIAM	✓
VAN JOHNSON, LI	✓
TIETZ, TIMO	✓
SCHULZ, JOHANN	✓
SCHMIDT, ANNE	✓
RASCH, JUERGEN	✓
QUERFELD, ELA	✓
PONEL, ANTJE	✓
PAPE, JENS	✓
MONKA, MATTHIAS	✓
MAURISCHAT, VAN	✓

Figure 29 C2.2: Old index page aspect



Veranstaltung hinzufügen

Name:	
Start:	Datum: Heute Zeit: Jetzt
End:	Datum: Heute Zeit: Jetzt
Künstler:	
Location:	
Organizer:	

Figure 30 C2.3 : Old view 'Veranstaltung'

18.4. Appendix C: Use Cases-Actor mapping

General

Table 5 Use Cases - Actor mapping

Requirements (User Stories)	Use Cases-Actor
Display a clear list of songs that were/are going to be played during a performance.	Display song list-EO
display for each song, its copyright owner.	Display copyright owner-EO
offer services in both German and English.	Select Language (English/German)-All the actors
provide a CSV file with payment data for each user on January 1st of each year.	Provide CSV of Payment Data
require users to list every concert they play in the GEMA online system.	List Concerts in GEMA System-Artist
delete all the data of a client.	Delete Client Data-EO

Performer

Requirements (User Stories)	Use Cases-Actor
Fill in a list of songs that I want to play at a certain date.	Register as Performer - Performer Login- Performer Register a song- Performer
Access the full lists of concerts I have registered in the past.	Login- Performer
See which performances have been sent to GEMA.	Login- Performer
See which performances have outstanding payments.	Login- Performer

Event organizer (EO)

Requirements (User Stories)	Use Cases-Actor
To fill in a list of songs that were/ will be played at a certain event.	Register as an event organizer-EO Login-EO Register Event-EO Register Track list-EO Generate GEMA form-EO (Will be sent to GEMA, with the Publishing representative as the actor)
See how much I need to pay in royalties, per performance.	Login-EO See Outstanding bills-EO

Author

Requirements (User Stories)	Use Cases-Actor
Register an account so that I have easy access to my content.	Register as Author-Author
Easily fill in information about my copyrighted content.	Login-Author
Access a list of performers that have played / are going to play my copyrighted content.	Login-Author

Publishing agency Representative (KR)

Requirements (User Stories)	Use Cases-Actor
have an overview of the most recent performances and their statuses, which payments have been fulfilled, and what their status is with GEMA.	See the summary of the most recent performances-KR
press a button that would print the difference in the list of events – between our system and GEMA's (comparing the CSV files).	See the list of event differences-KR
As an admin I want to see a list of registered authors.	See the list of registered authors-KR
For each author, I want to be able to see their information.	See the information of each author-KR
As an admin I want to see a list of registered performances.	See the tracklist of each event-KR
For each author, I want to change their information.	Edit Author Information-KR
As an admin, I want to search a specific artist by their name.	Search the artists by their name-KR

For each performance, I want to see their location, start date, and end date.	See all events an artist registered to-KR
As an admin I want to see a list of track list-***Musikfolgen***.	See the tracklist of each event-KR
As an admin I want to see a list of bills.	See the information of an event-KR See the list of bills-KR

18.5. Appendix D: Security by Design table - Prepared by: Dipti K. Sarmah

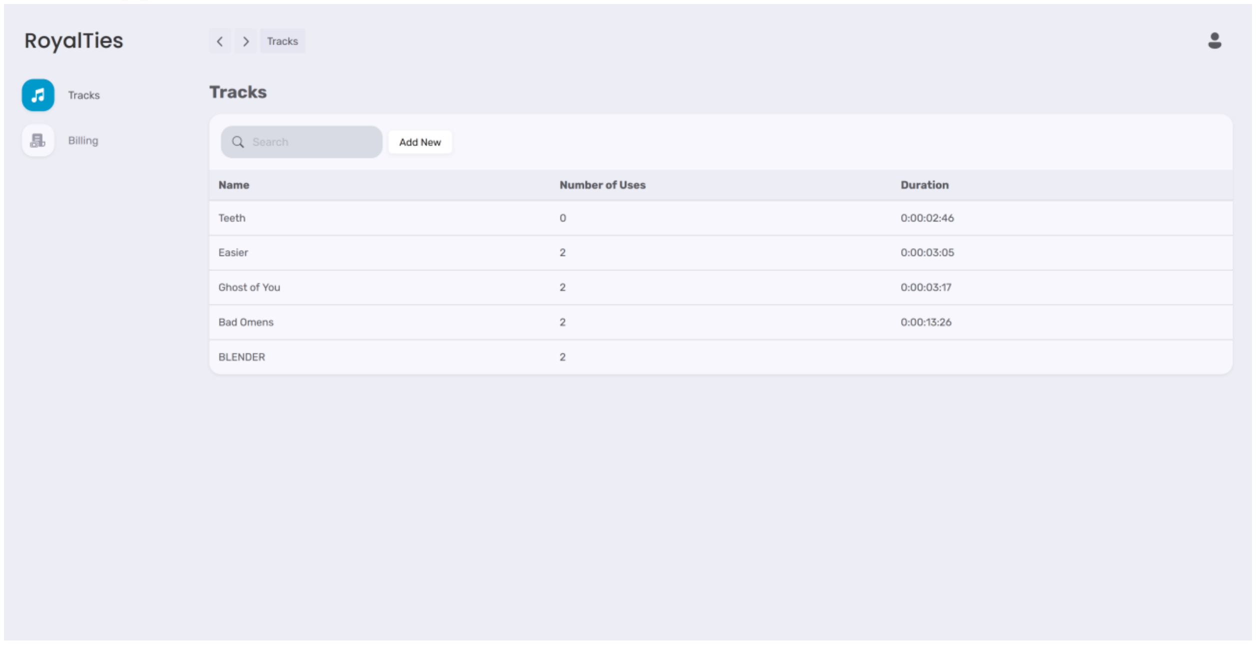
Table 6 Security By Design Checklist Table

Security Policy	**Confidentiality, Integrity, and Availability**					
Security Requirements	**Security mechanisms (List down for your application)**	**Remarks on why you considered these requirements? (in a brief)**	**Supplement requirements for your application (user story/Abuse case)**	**Risk identification/Threat Assessment (at least one risk identification/abuse case)**	**Appropriate Security Controls**	**Tick ✓if you have applied the given security controls as suggested in the left column**
Authentication	Checking password	For granting access to multiple users and for users to have their individual profiles, we need to authenticate their	- Use case: As a user, I can enter my user name and passwords to access the application. - Abuse Case: As an attacker, I can enter the default passwords to access the	- The length of the password is less than 4 characters. - The password is not strong enough. - A wrong password is entered more	- The system verifies that there are no default passwords used by the application or any of its components. - The default policy on endpoints is to	✓

		username with a password.	application.	than 5 times.	require an authenticated user with an admin rank (highest security by default, so that if forgotten, it will refuse access to some users, instead of accidentally providing access to more users). - All passwords are hashed and salted before being stored. - Password requires at least 6 characters, requires non-alphanumeric characters, lowercase and uppercase characters and at least one digit. - Passwords are entered on the separate accounts webapp, subsequent requests to an API use JWT tokens (possibly encrypted) to communicate authentication/authorization details.	

Authorization	Access control policies- User based.	For granting access to limited sources.	- Use case: As a user, I can enter my user name and passwords to access the application. - Abuse Case: As an attacker, I can enter the default passwords to access the application.	Insecure passwords can lead to unauthorized access.	Application uses policy based authorization. Currently policies RequireAdmin, RequireArtist, RequireOrganiser, and the default policy of an authenticated user. The admin role is authorized in all these policies.	✓
Audit	Protection of Log files by backup	For allowing the user to see a saved log containing information on errors encountered by the application and general logs such as user login activity.	- Use case: As a user, I can see information about my past activity, warnings and errors in a log file. - Abuse Case: As an attacker, I can gain access to the log file and get information about the user's activity.	An attacker could access sensitive information about the user.	Log files have (linux) file permissions to only be read/written by the web application user/nginx system user. The nginx static file serving user does not have permission to these files.	✓
Team members' reviewed:	(David, Yes) (Brianna, Yes) (Michiel, Yes) (Sebastian, Yes) (Radhmer, Yes)					

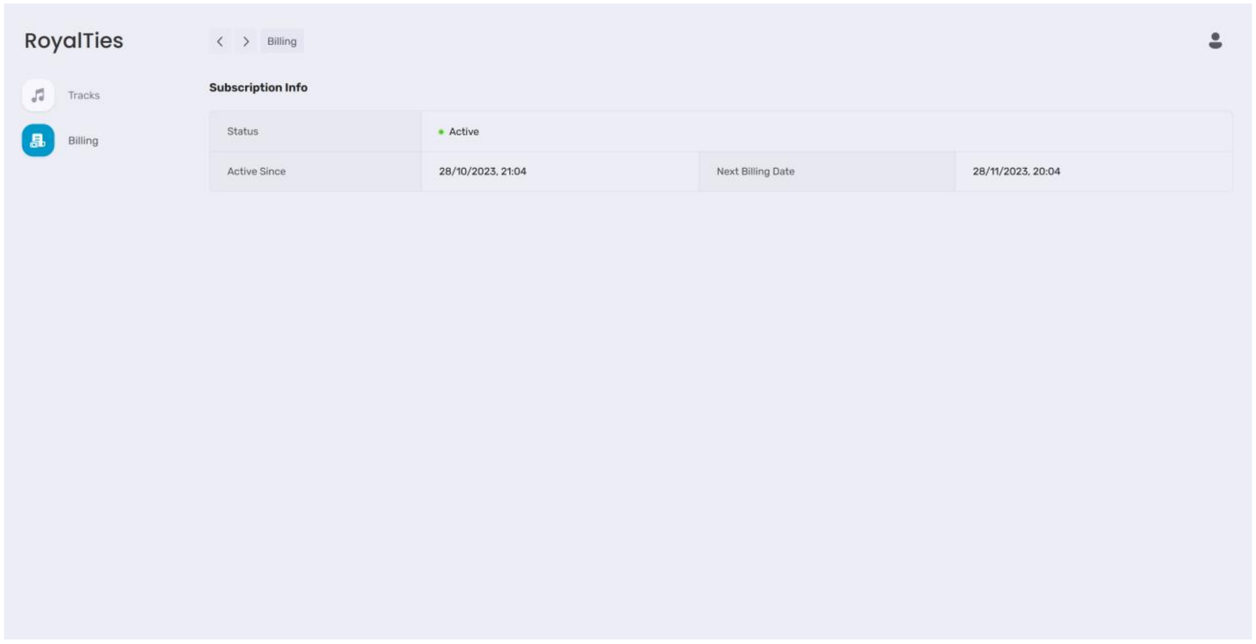
18.6. Appendix E: New UI



Tracks

Name	Number of Uses	Duration
Teeth	0	0:00:02:46
Easier	2	0:00:03:05
Ghost of You	2	0:00:03:17
Bad Omens	2	0:00:13:26
BLENDER	2	

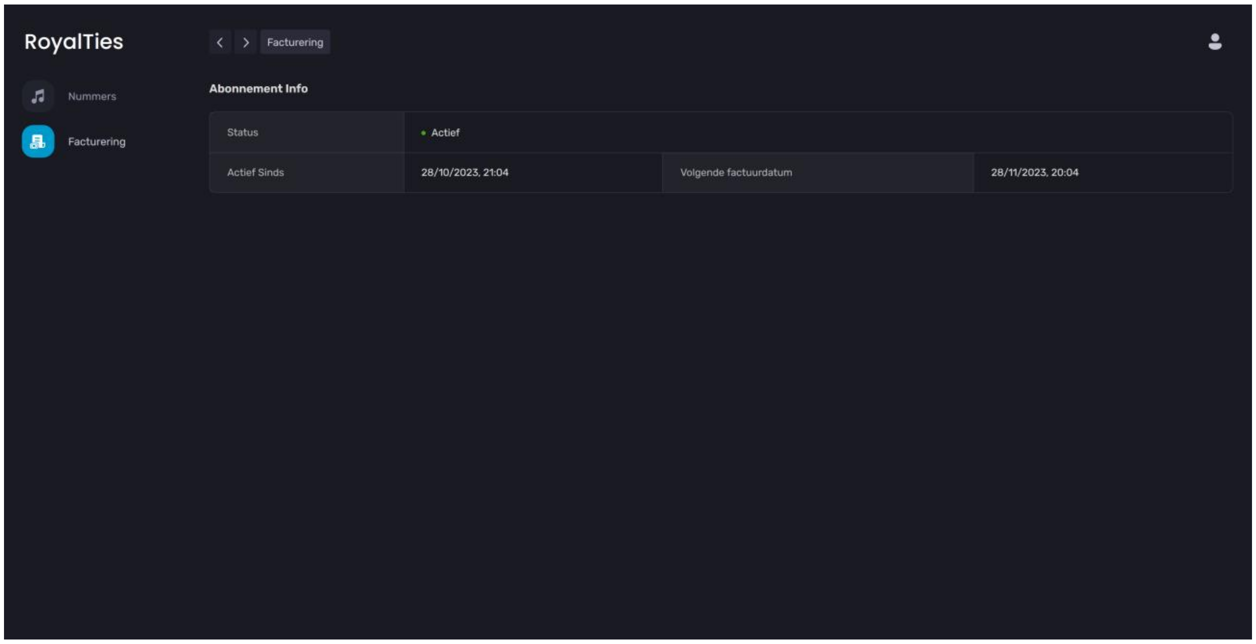
Figure 31 Artist track index screen



Subscription Info

Status	● Active		
Active Since	28/10/2023, 21:04	Next Billing Date	28/11/2023, 20:04

Figure 32 Active billing screen



RoyalTies

Facturering

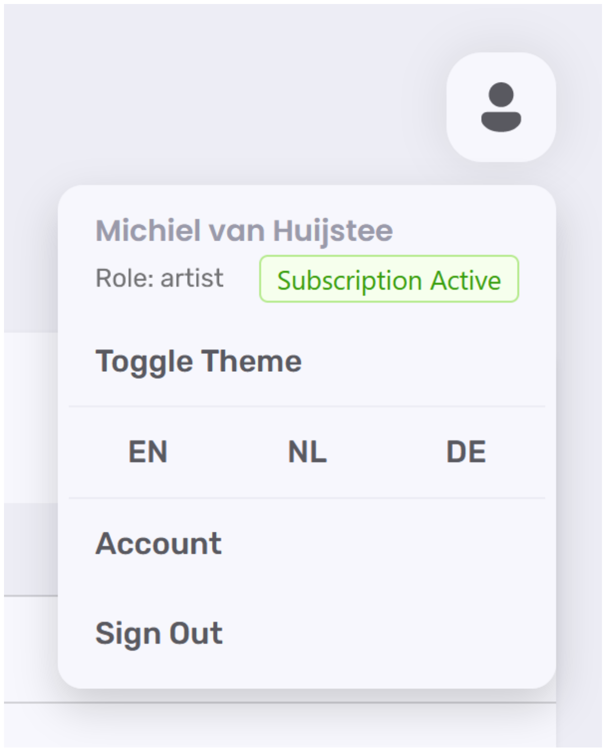
Nummers

Facturering

Abonnement Info

Status	● Actief		
Actief Sinds	28/10/2023, 21:04	Volgende factuurdatum	28/11/2023, 20:04

Figure 33 Active billing screen, but in Dutch and in dark mode



Michiel van Huijstee

Role: artist

Subscription Active

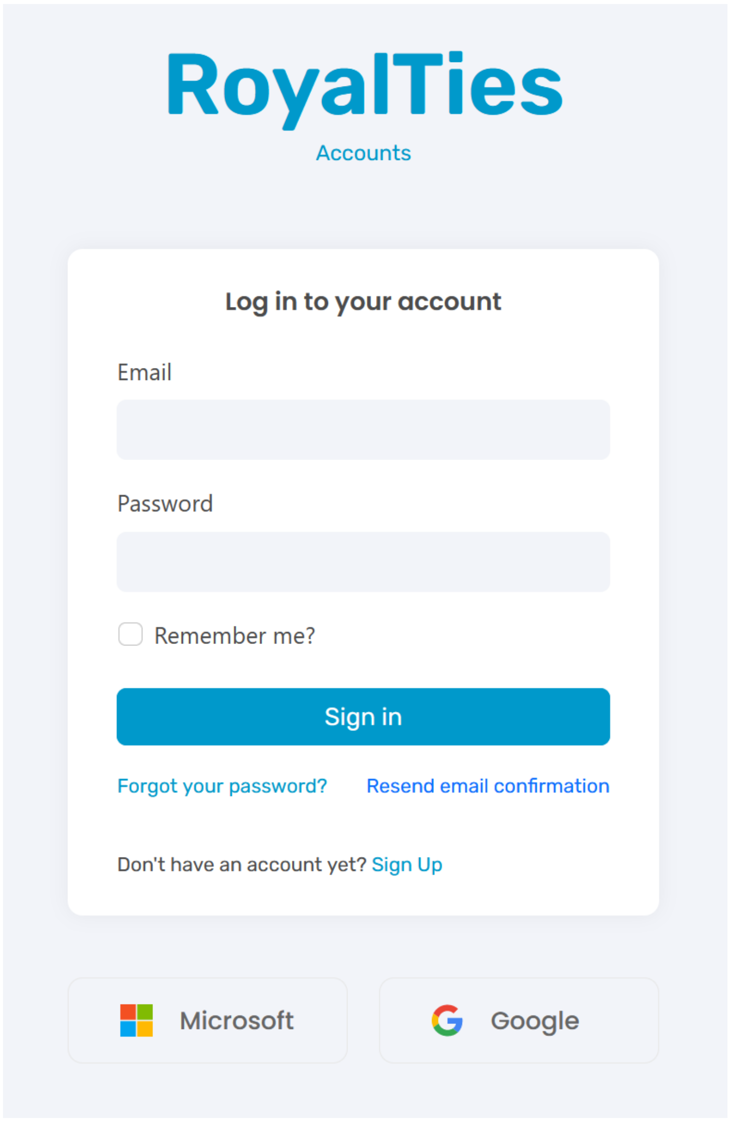
Toggle Theme

EN NL DE

Account

Sign Out

Figure 34 User account popover


RoyalTies

Accounts

Log in to your account

Email

Password

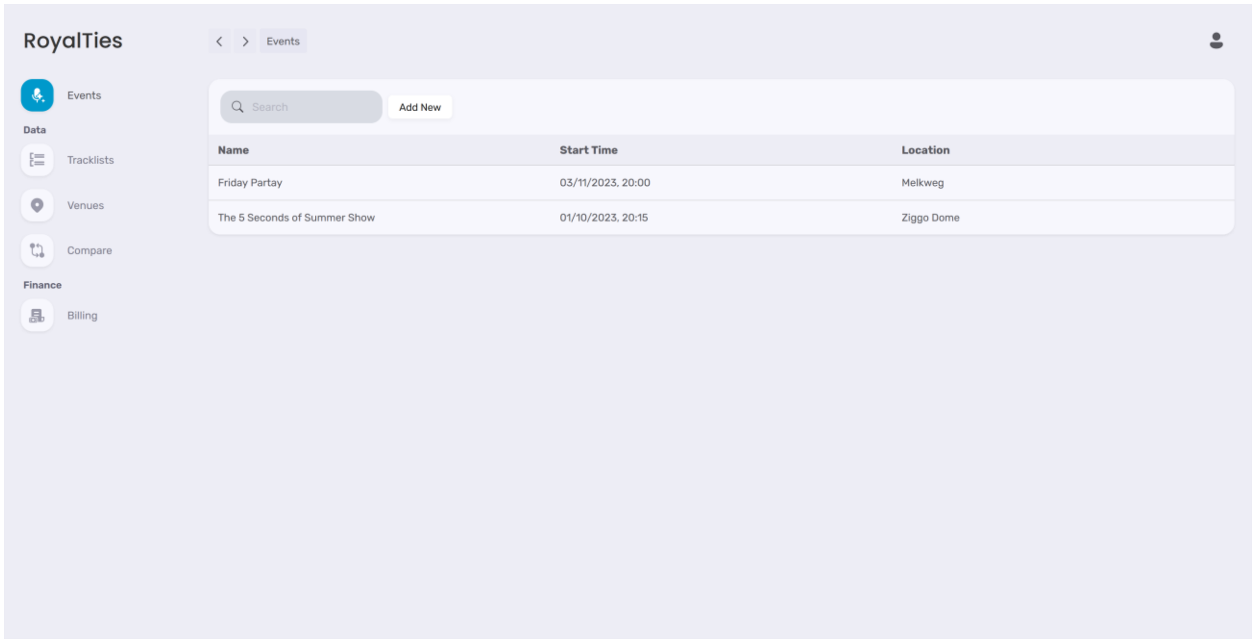
Remember me?

Sign in

Forgot your password? Resend email confirmation

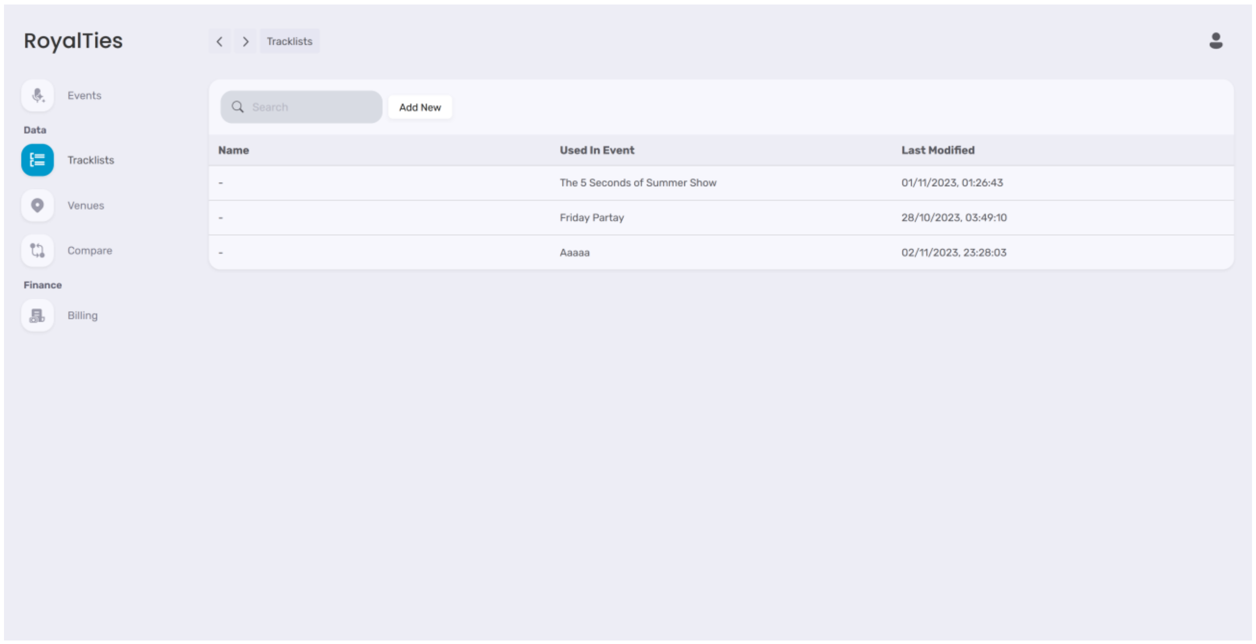
Don't have an account yet? Sign Up

Figure 35 Login Screen, sized appropriately for the size of a phone



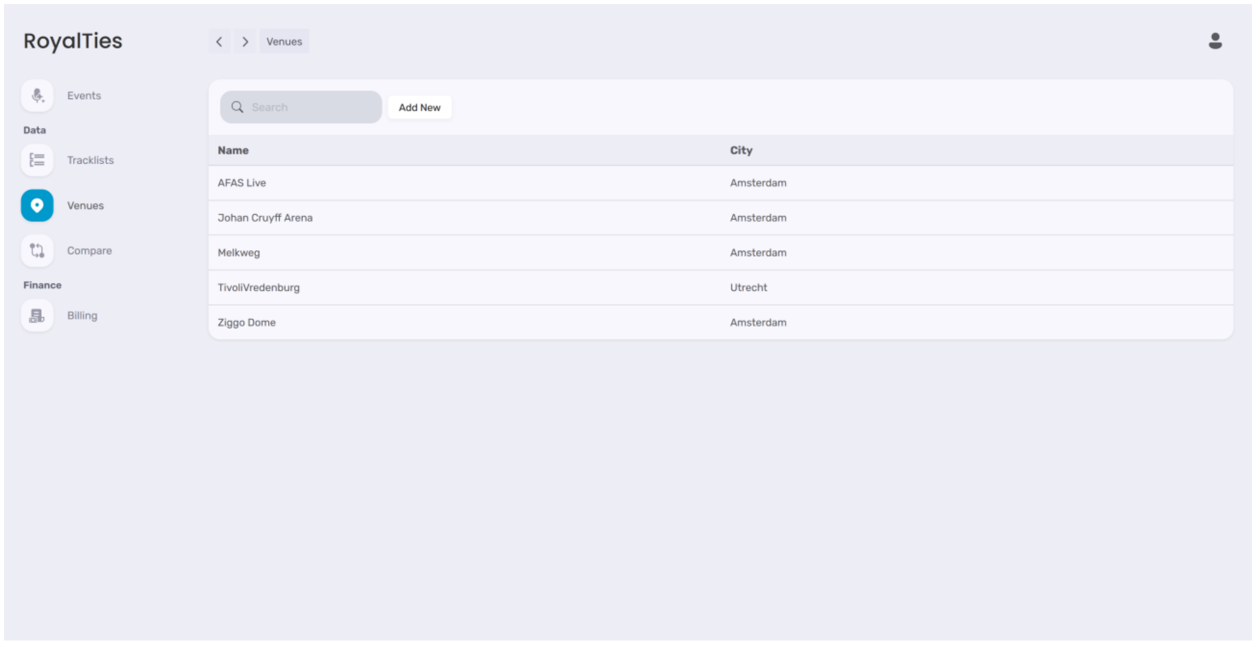
Name	Start Time	Location
Friday Partay	03/11/2023, 20:00	Melkweg
The 5 Seconds of Summer Show	01/10/2023, 20:15	Ziggo Dome

Figure 36 Event Dashboard



Name	Used In Event	Last Modified
-	The 5 Seconds of Summer Show	01/11/2023, 01:26:43
-	Friday Partay	28/10/2023, 03:49:10
-	Aaaaa	02/11/2023, 23:28:03

Figure 37 Track lists dashboard



Name	City
AFAS Live	Amsterdam
Johan Cruyff Arena	Amsterdam
Melkweg	Amsterdam
TivoliVredenburg	Utrecht
Ziggo Dome	Amsterdam

Figure 38 Venues Dashboard

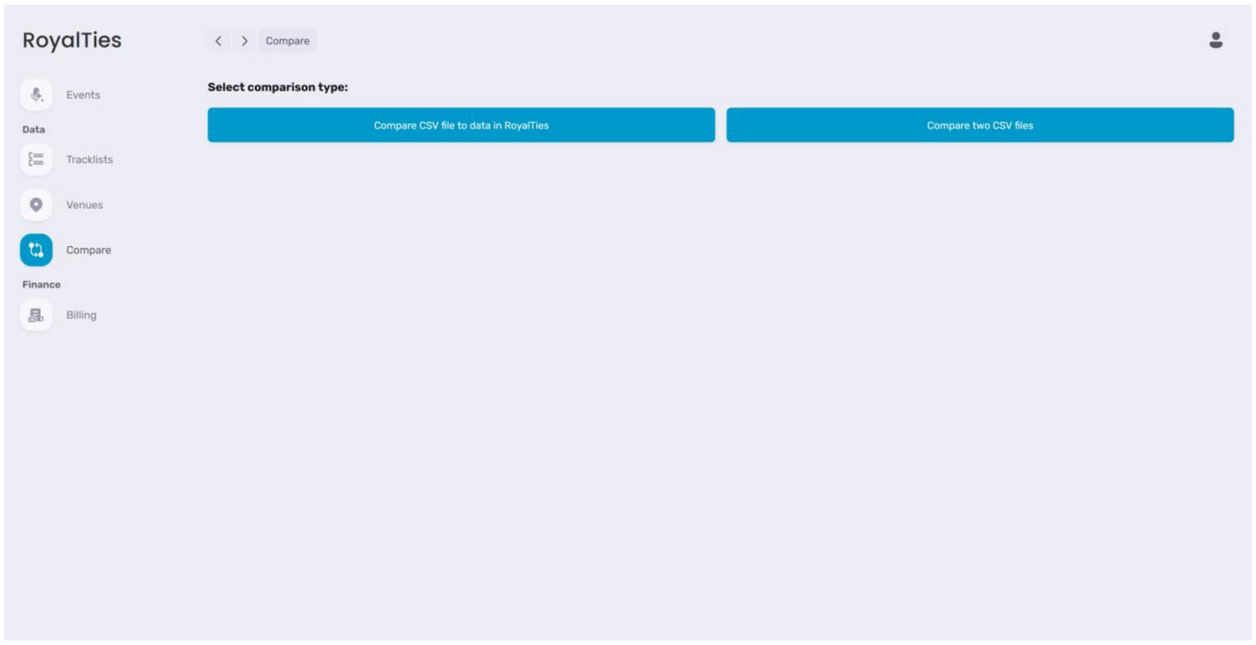


Figure 39 CSV Comparison Screen

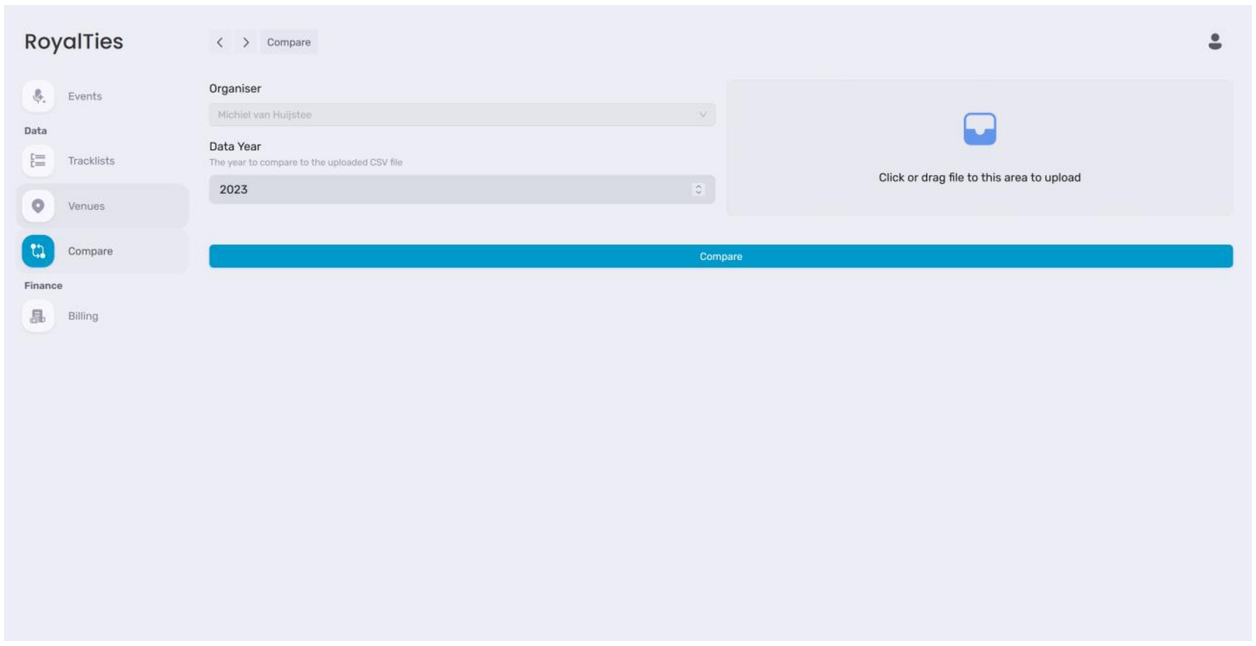


Figure 40 CSV Comparison Screen

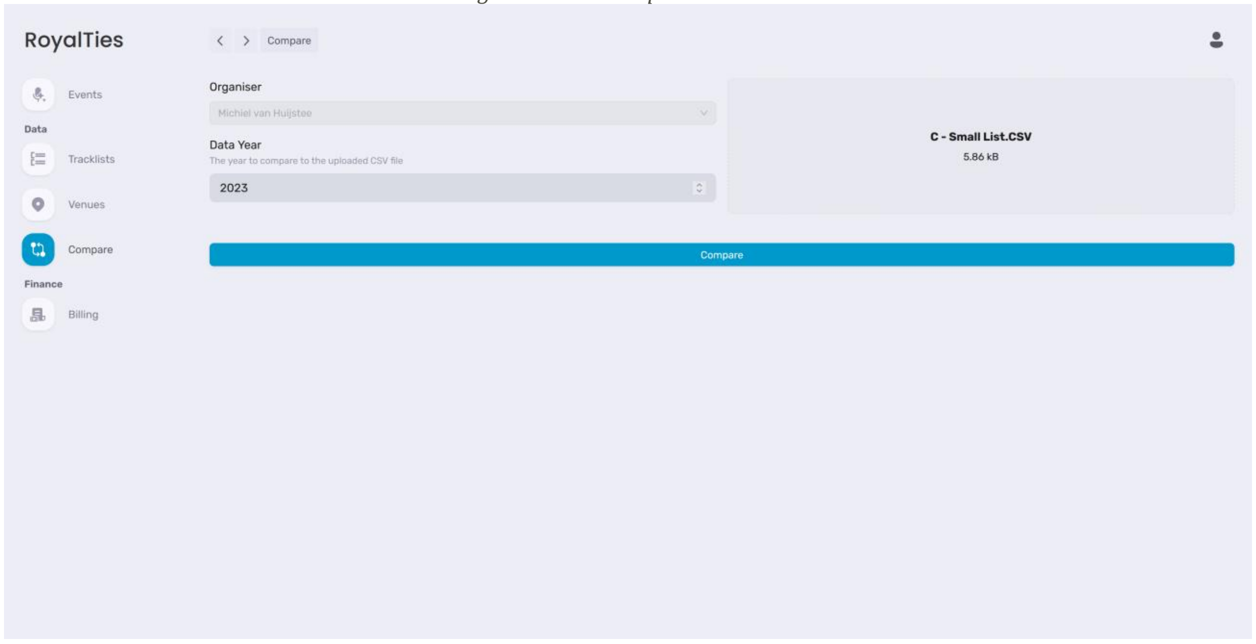


Figure 41 CSV Comparison Screen

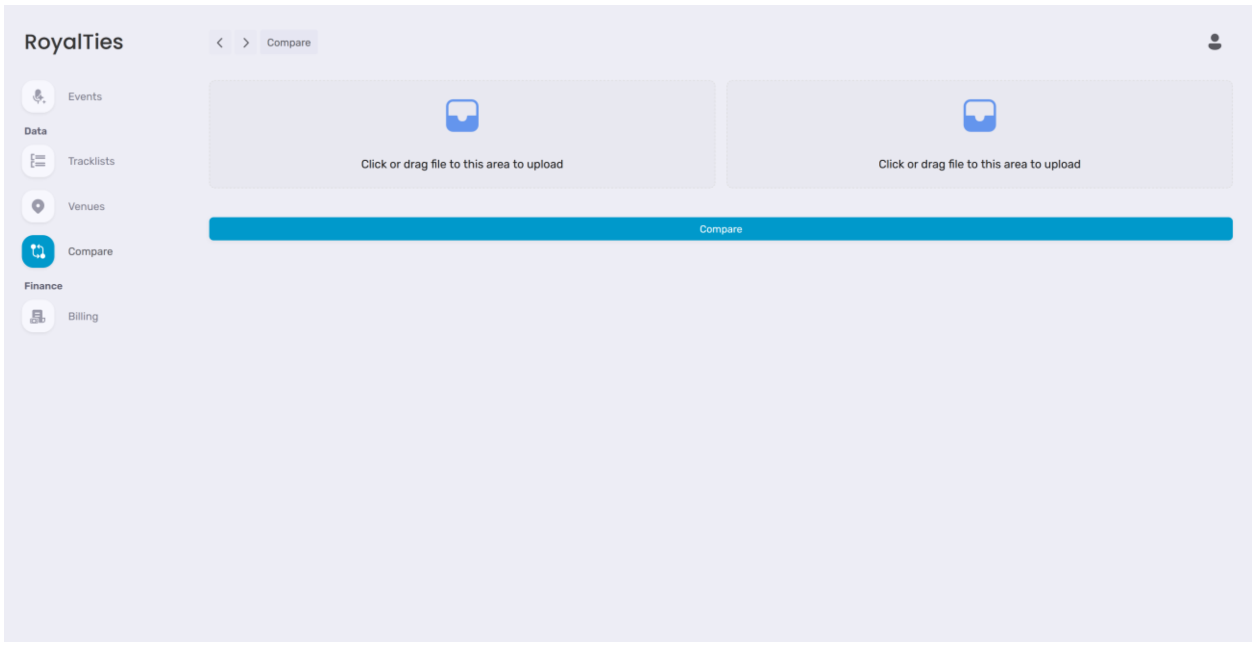
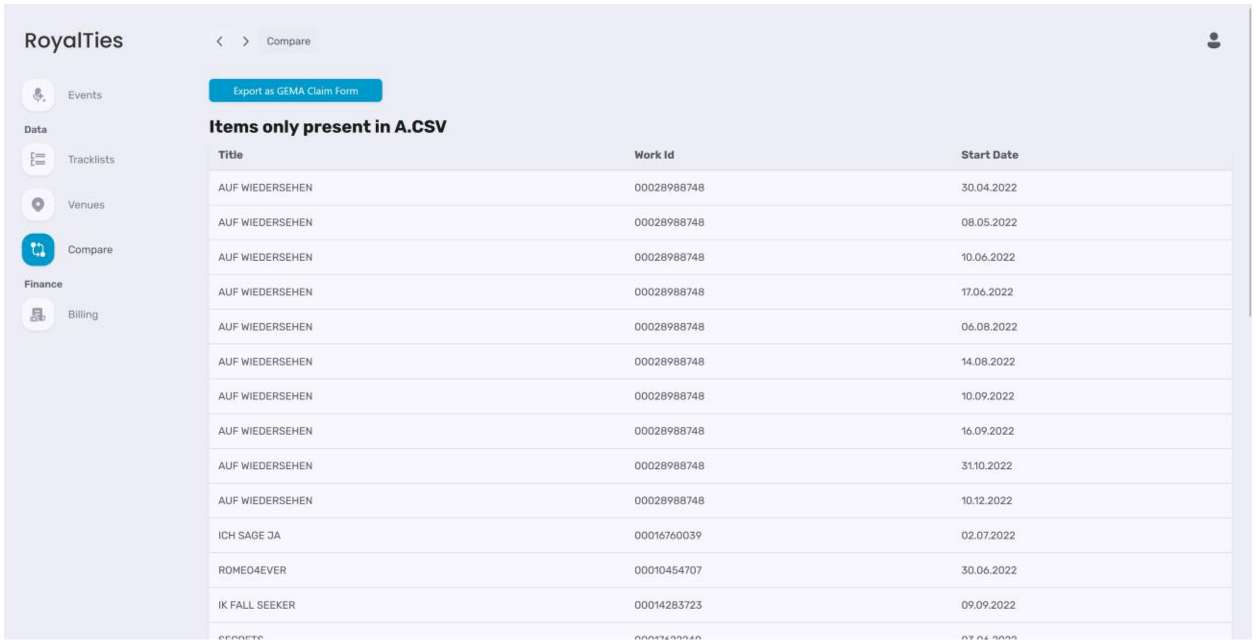

RoyalTies
Compare
Events
Data
Tracklists
Venues
Compare
Finance
Billing
Click or drag file to this area to upload
Click or drag file to this area to upload
Compare

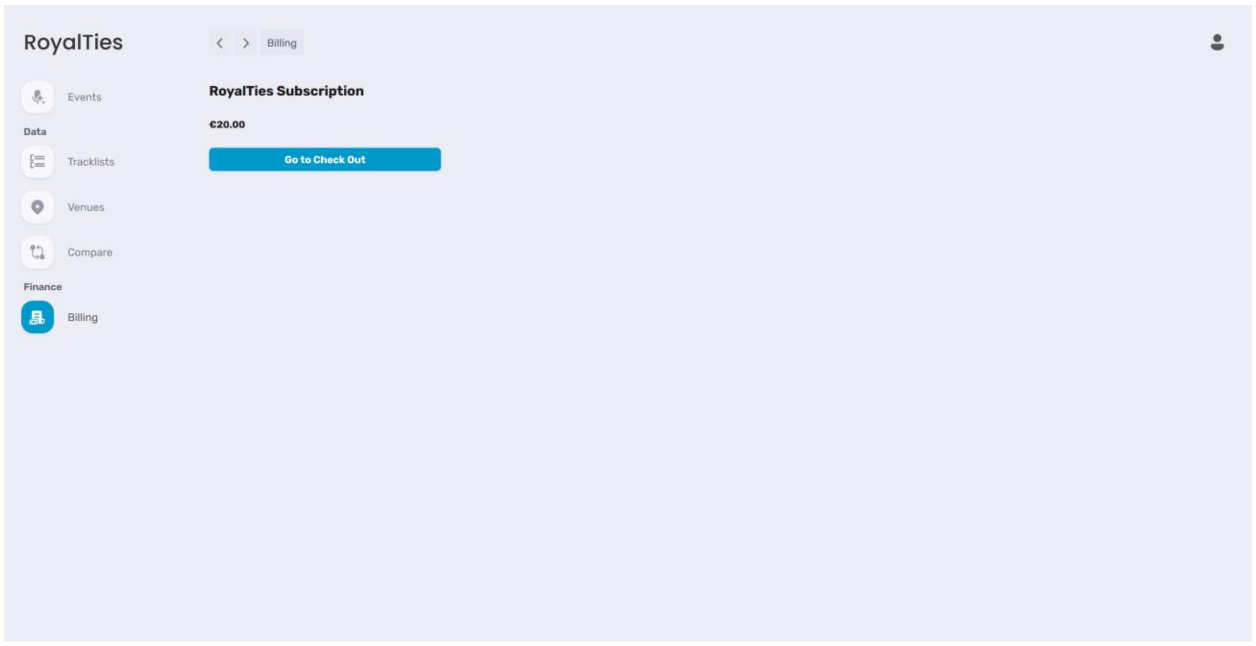
Figure 42 CSV Comparison Screen



Items only present in A.CSV

Title	Work Id	Start Date
AUF WIEDERSEHEN	00028988748	30.04.2022
AUF WIEDERSEHEN	00028988748	08.05.2022
AUF WIEDERSEHEN	00028988748	10.06.2022
AUF WIEDERSEHEN	00028988748	17.06.2022
AUF WIEDERSEHEN	00028988748	06.08.2022
AUF WIEDERSEHEN	00028988748	14.08.2022
AUF WIEDERSEHEN	00028988748	10.09.2022
AUF WIEDERSEHEN	00028988748	16.09.2022
AUF WIEDERSEHEN	00028988748	31.10.2022
AUF WIEDERSEHEN	00028988748	10.12.2022
ICH SAGE JA	00016760039	02.07.2022
ROMEO4EVER	00010454707	30.06.2022
IK FALL SEEKER	00014283723	09.09.2022

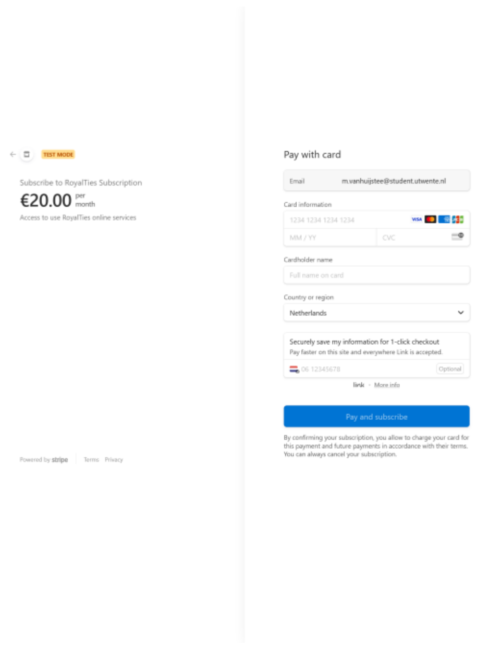
Figure 43 CSV Comparison Screen



RoyalTies Subscription

€20.00

Figure 44 Inactive billing screen



Subscribe to RoyalTies Subscription

€20.00 per month

Access to use RoyalTies online services

Pay with card

Email m.vanhuijstee@student.utwente.nl

Card information

1234 1234 1234 1234

MM / YY

CVC

Cardholder name

Full name on card

Country or region

Netherlands

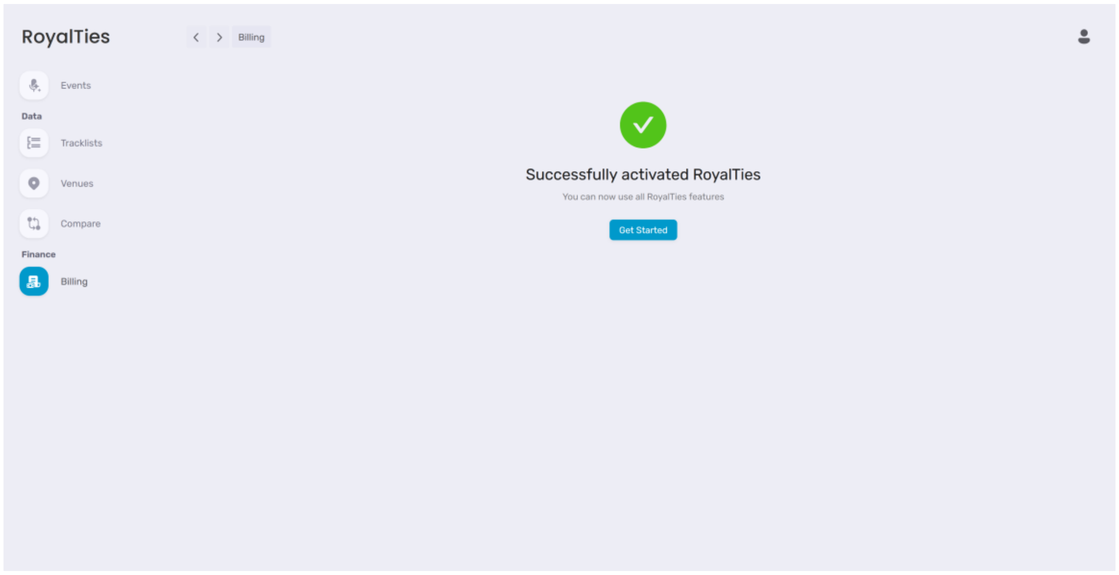
Securely save my information for 1-click checkout

Pay faster on this site and everywhere Link is accepted.

06 12345678 Optional

By confirming your subscription, you allow to charge your card for this payment and future payments in accordance with their terms. You can always cancel your subscription.

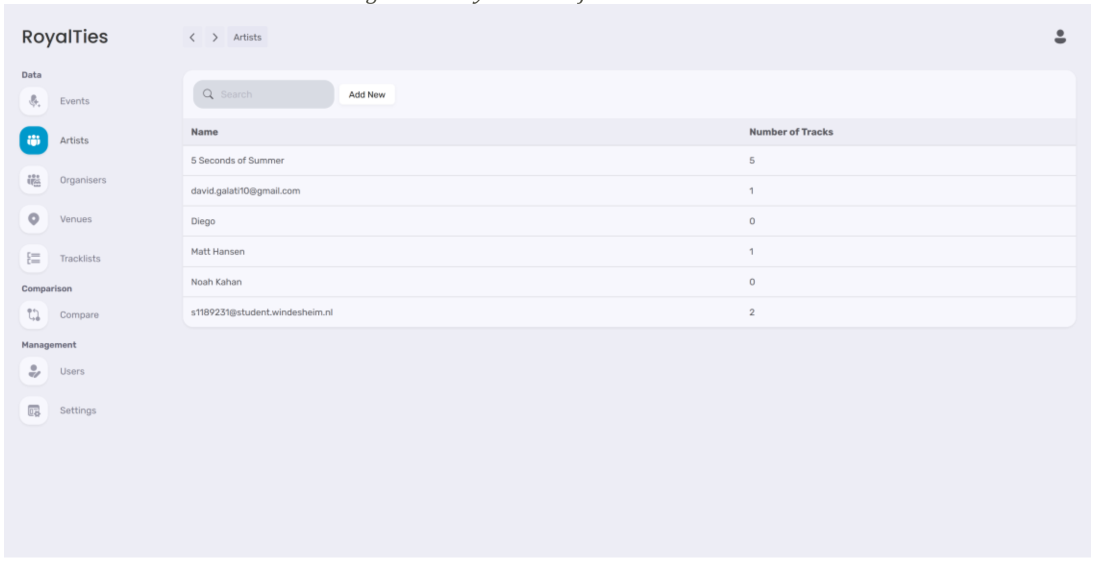
Figure 45 Stripe Payment Portal



Successfully activated RoyalTies

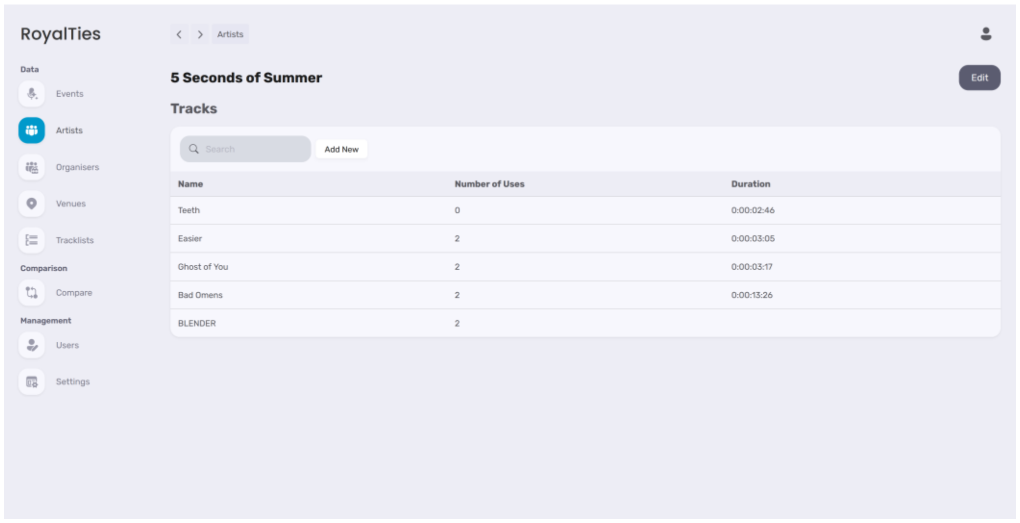
You can now use all RoyalTies features

Figure 46 Payment Confirmation Screen



Name	Number of Tracks
5 Seconds of Summer	5
david.galati10@gmail.com	1
Diego	0
Matt Hansen	1
Noah Kahan	0
s1189231@student.windesheim.nl	2

Figure 47 Artist Management Dashboard

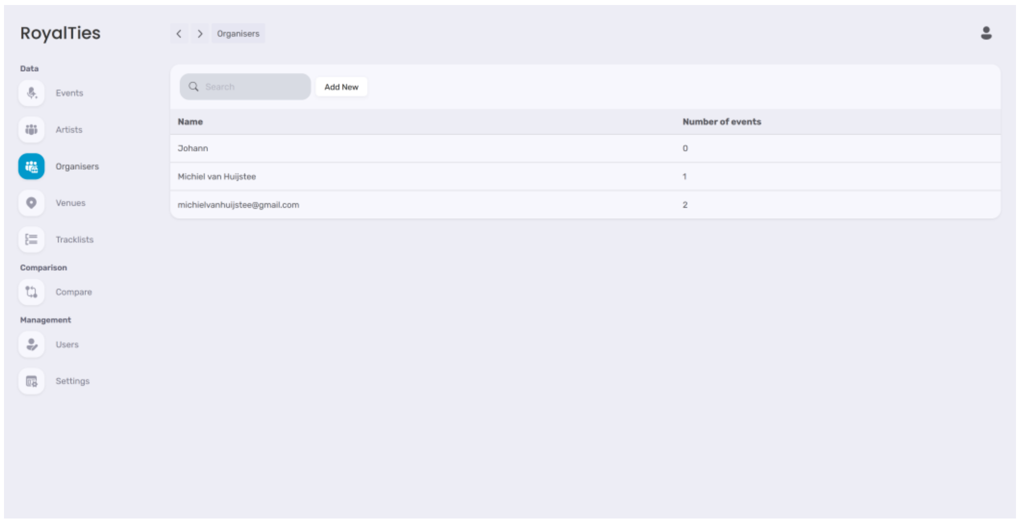


5 Seconds of Summer

Tracks

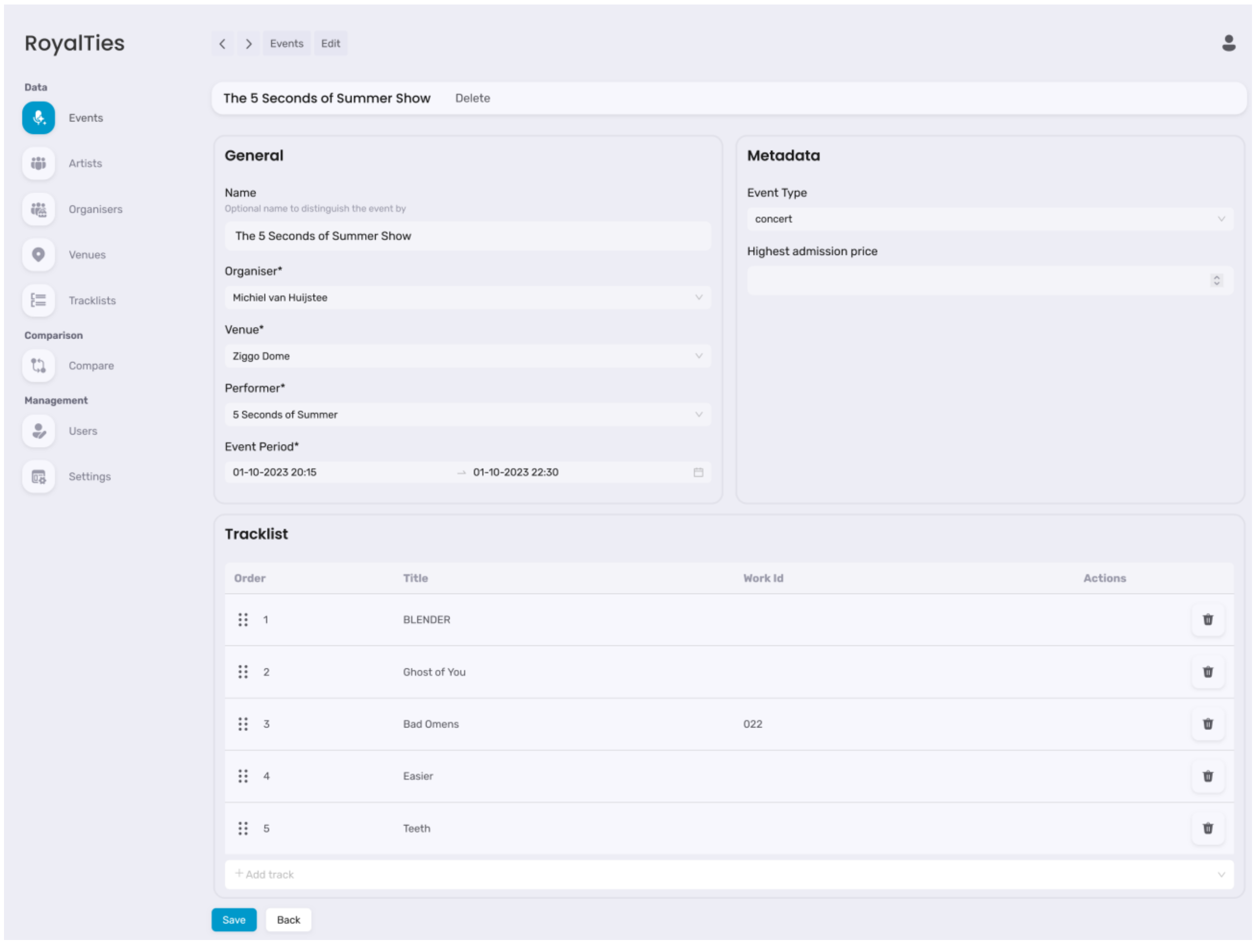
Name	Number of Uses	Duration
Teeth	0	0:00:02:46
Easier	2	0:00:03:05
Ghost of You	2	0:00:03:17
Bad Omens	2	0:00:13:26
BLENDER	2	

Figure 48 Artist Management Dashboard



Name	Number of events
Johann	0
Michiel van Huijstee	1
michielvanhuijstee@gmail.com	2

Figure 49 Organizer Management Dashboard



The 5 Seconds of Summer Show

General

Name
Optional name to distinguish the event by
The 5 Seconds of Summer Show

Organiser*
Michiel van Huijstee

Venue*
Ziggo Dome

Performer*
5 Seconds of Summer

Event Period*
01-10-2023 20:15 → 01-10-2023 22:30

Metadata

Event Type
concert

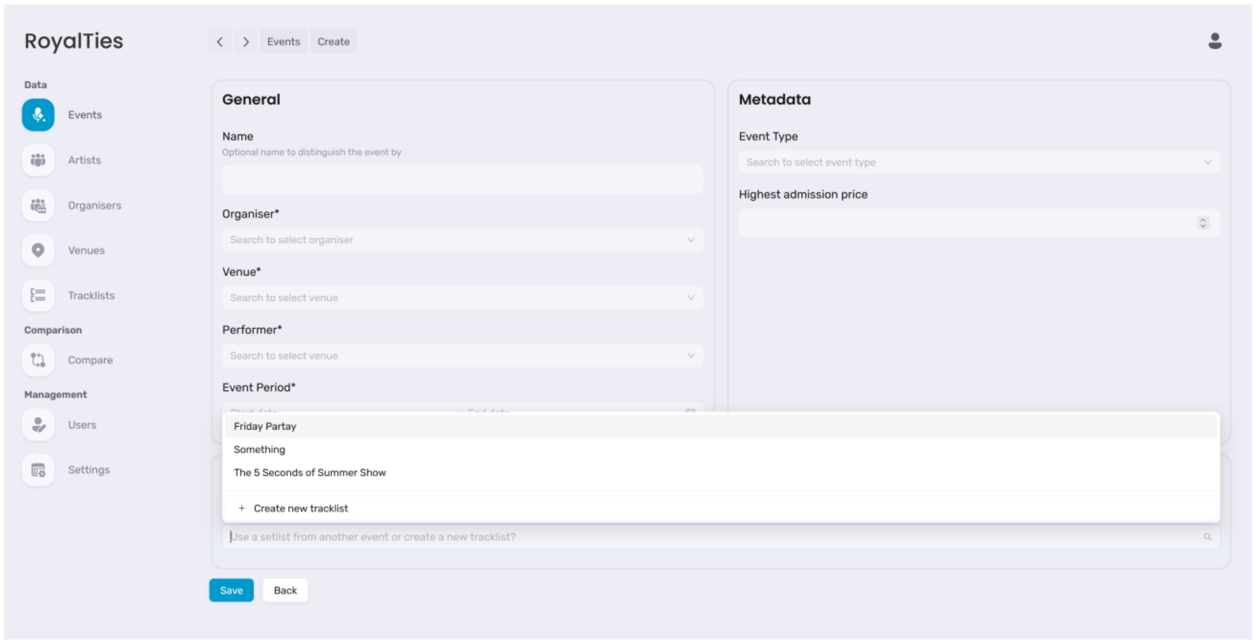
Highest admission price

Tracklist

Order	Title	Work Id	Actions
1	BLENDER		
2	Ghost of You		
3	Bad Omens	022	
4	Easier		
5	Teeth		

+ Add track

Figure 50 Event editing screen



General

Name
Optional name to distinguish the event by

Organiser*
Search to select organiser

Venue*
Search to select venue

Performer*
Search to select venue

Event Period*

Metadata

Event Type
Search to select event type

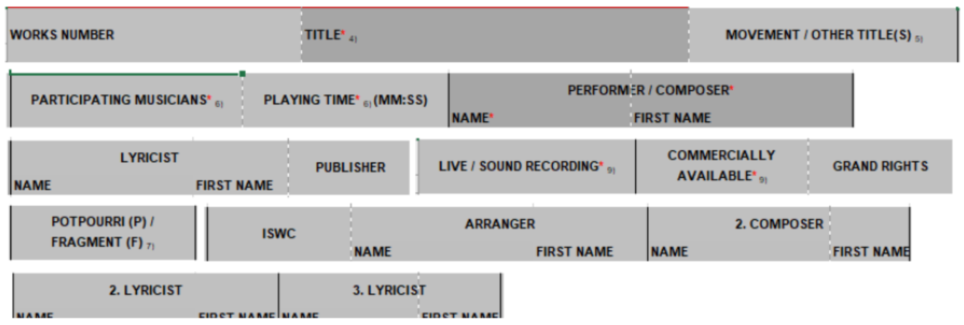
Highest admission price

Friday Partay
Something
The 5 Seconds of Summer Show
+ Create new tracklist

Use a setlist from another event or create a new tracklist?

Figure 51 Event creation screen

18.7. Appendix F: GEMA form



WORKS NUMBER	TITLE* 4)	MOVEMENT / OTHER TITLE(S) 5)

PARTICIPATING MUSICIANS* 6)	PLAYING TIME* 6) (MM:SS)	PERFORMER / COMPOSER*	
		NAME*	FIRST NAME

LYRICIST		PUBLISHER	LIVE / SOUND RECORDING* 9)	COMMERCIALLY AVAILABLE* 9)	GRAND RIGHTS
NAME	FIRST NAME				

POTPOURRI (P) / FRAGMENT (F) 7)	ISWC	ARRANGER		2. COMPOSER	
		NAME	FIRST NAME	NAME	FIRST NAME

2. LYRICIST		3. LYRICIST	
NAME	FIRST NAME	NAME	FIRST NAME

Figure 52 Track list individual information form for each song

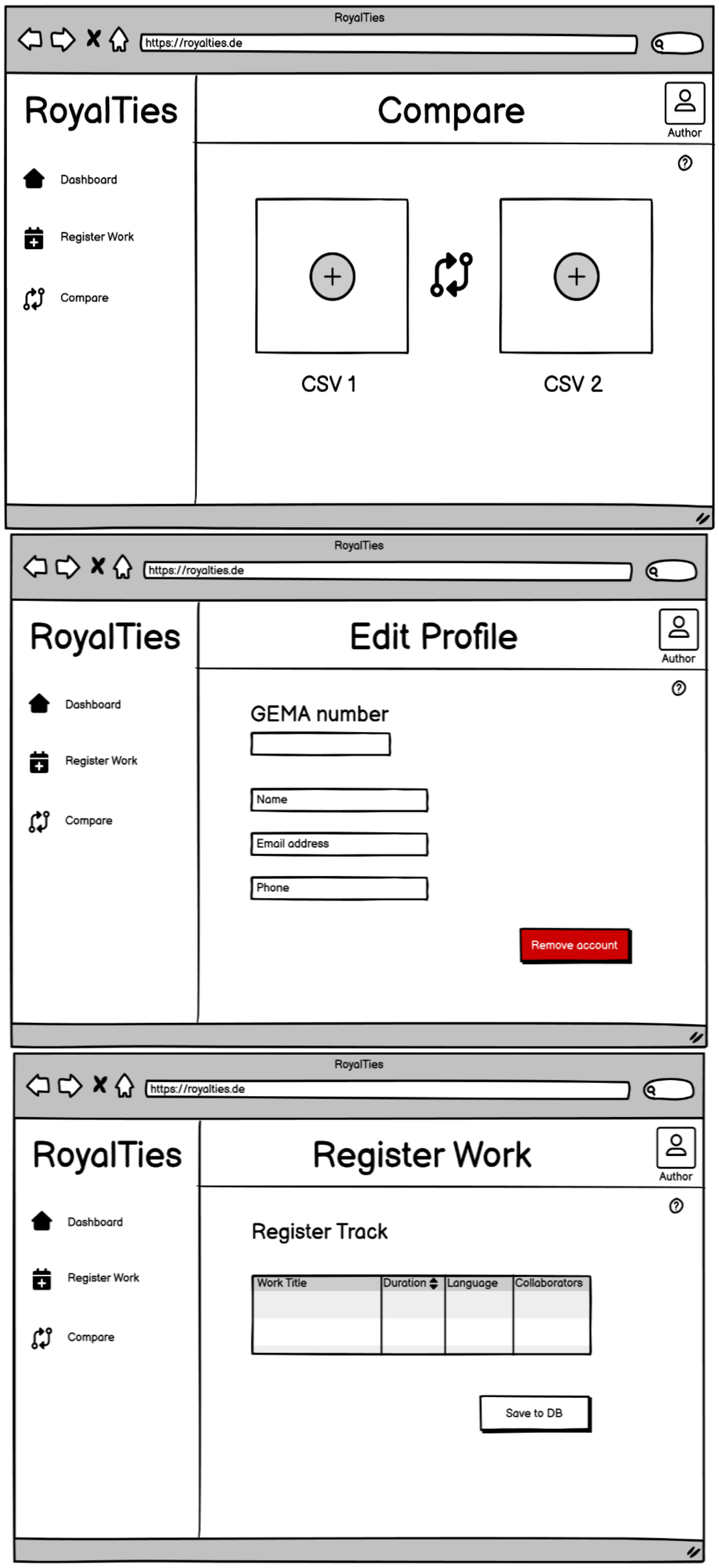


Figure 53 Balsamiq Wireframes