VRIJE
UNIVERSITEIT
BRUSSEL

PROJECT REPORT

Web Technologies

Reinout Cloosen, Vic De Hondt, Dante Tibollo
Group 5

December 2023

Science and Bio-Engineering Sciences

Contents

1 Introduction

This project report provides the reader with all information regarding our web application. We created the application "Concerto".

The goal of Concerto is to make it easier finding people with the same interest about concerts. When you create an account, on the homepage you are shown personalized events based on your favourite genres. Users can see the rating of the artist and the venue of an event, and for a detailed look visit the rating page where reviews from other users are posted.

The search system allows users to search for events by name and filter at the same time. Privacy has been considered too: you can choose who sees your information.

Upcoming sections of this document are dedicated to explaining the implementation in detail. The second section describes all the functionalities. The third and fourth section explain the decisions made on the frontend and backend. Appendix B contains the timesheets of each person working on the project.

2 Implemented requirements

This section is dedicated to explaining the functional requirements that were implemented in our project. For all functionality both frontend and backend are described. We were able to finish all goals we stated at the beginning of our project.

2.1 Users

2.1.1 Registering

Users are able to create an account on our web application. Below a listing of all the necessary information is given.

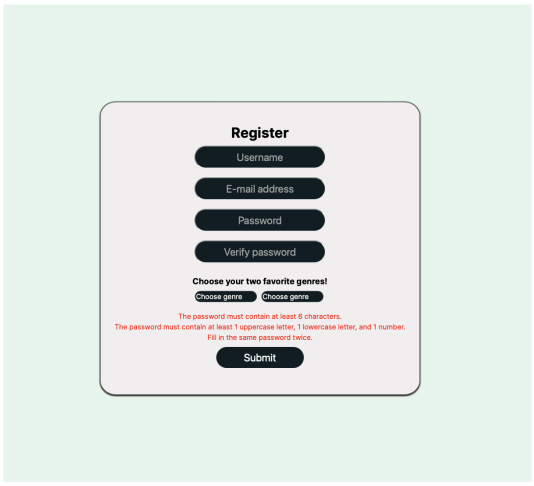
- username*
- email*
- password
- two music genres

Fields which are marked with "*" are unique. Figure 1 shows the form a user sees when registering. A user is only able to choose two genres because we had issues with storing a dynamic amount of genres on the backend. Otherwise we would have allowed the user to select an unrestricted amount of genres.

A provided password should have at least 6 characters, a small letter, a capital letter and a number.

2.1.2 login

Login requires the username and password of a user. There are 3 possibilities checked in this order:



The password must contain at least 6 characters.
The password must contain at least 1 uppercase letter, 1 lowercase letter, and 1 number.
Fill in the same password twice.

Figure 1: Registration form

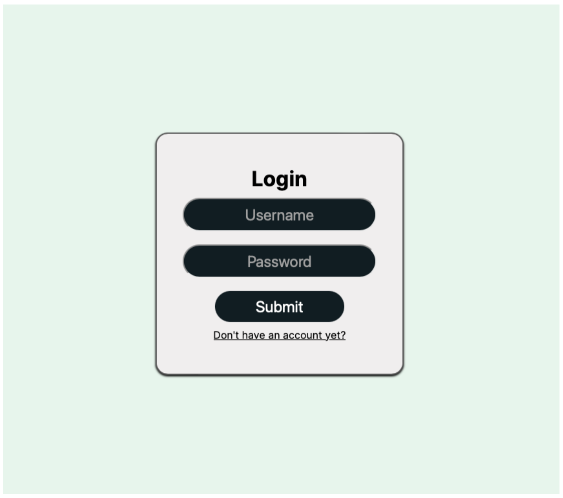


Figure 2: Login form

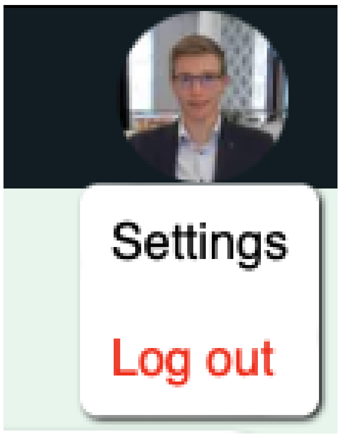


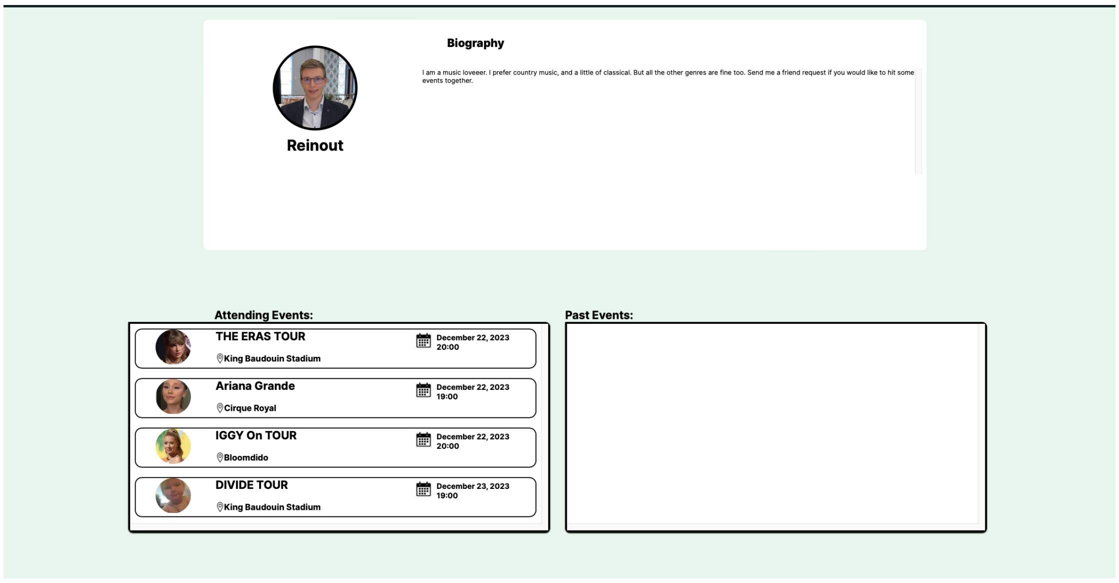
Figure 3: Logout button

1. There exists no user with this username.
2. The user exists and the password is incorrect.
3. The user exists and the password is correct.

The first two situations will provide the user with an appropriate error indicating what went wrong. The third situation leads to a successful login. Figure 2 shows the form a user uses to login.

2.1.3 Logout

When a user is logged in, his profile picture will appear at the most right position in the navigation bar. When the user has no profile picture a default icon is shown. When hovering over his picture a drop-down menu provides him with the option to log out. Figure 3 shows the log out button.


Reinout

Biography

I am a music loveeer. I prefer country music, and a little of classical. But all the other genres are fine too. Send me a friend request if you would like to hit some events together.

Attending Events:

THE ERAS TOUR	King Baudouin Stadium	December 22, 2023 20:00
Ariana Grande	Cirque Royal	December 22, 2023 19:00
IGGY On TOUR	Bloomdido	December 22, 2023 20:00
DIVIDE TOUR	King Baudouin Stadium	December 23, 2023 19:00

Past Events:

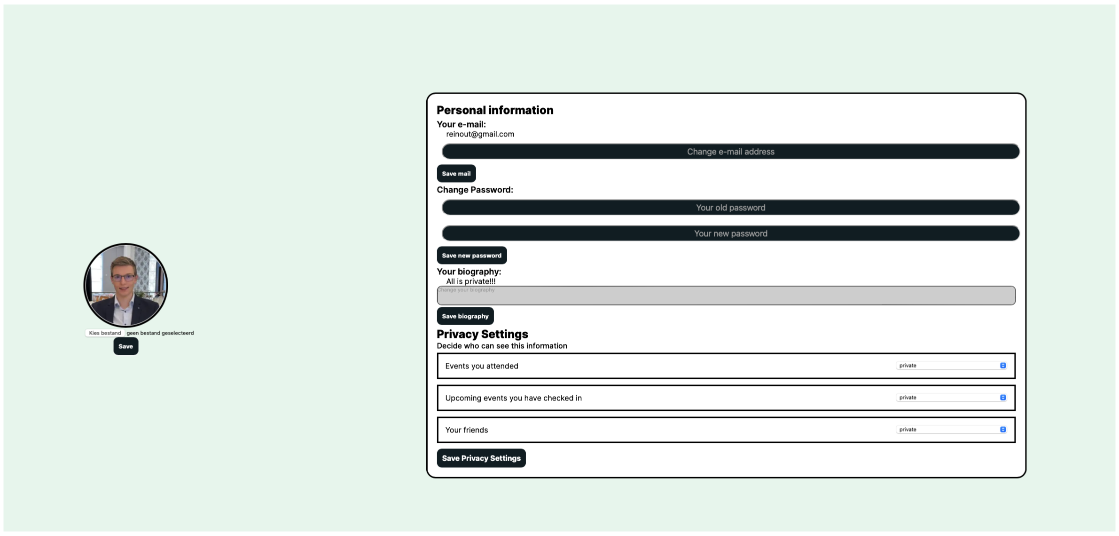
Figure 4: The account page of a user

2.1.4 Account information

For each user his profile can be displayed. This page shows the profile picture, username, a short biography and potentially the upcoming and past events of a user (if the viewer has permission to see this information). It can be seen in figure 4.

2.1.5 User settings

The settings page allows the user to edit his own profile when logged in. He can change his profile picture, e-mail, password, biography and privacy settings. If the user wants to change his password, in addition to giving his new password, he will also have to give his old password. This is done for security reasons. The new password must meet the requirements. The privacy settings let the user choose whether all his friends, everyone or no one can see his past events, the events he is checked in for and his friends. The settings page can be seen in figure 5.


Personal information

Your e-mail:
reinout@gmail.com

Change e-mail address

Save mail

Change Password:

Your old password

Your new password

Save new password

Your biography:
All is private!!!

Change your biography

Save biography

Privacy Settings

Decide who can see this information

Events you attended	private
Upcoming events you have checked in	private
Your friends	private

Save Privacy Settings

Figure 5: The settings page

2.2 Form validation

Form validation is performed on all forms, both server and client-sided.

On the registration page all the fields have to be filled in before sending a request. Real-time information about the passwords is provided to the user. This includes: if the length is okay, if it contains all the required characters and if both passwords match.

When a request is sent to the back-end there will be a verification if the email and username are not used by another user.

2.3 Core information

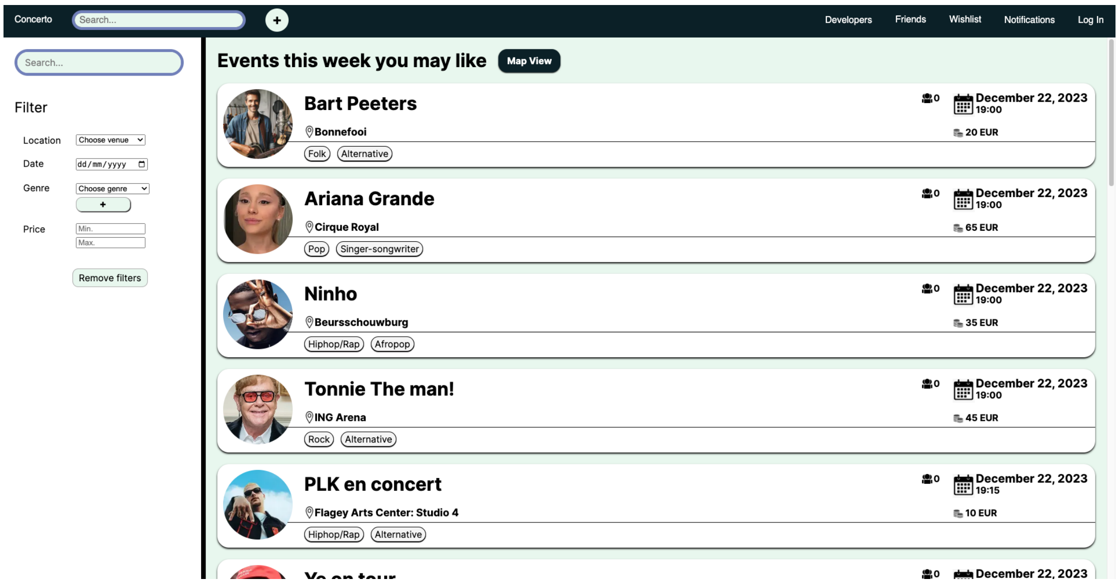
2.3.1 Creating and editing events

Users are able to add new and edit existing events. A user can only perform these actions when logged in. For adding an event we created a specific page were the user needs to fill in all the required information. When not all required fields are filled in, the user will get a warning when trying to create the event. We added a preview card of the event the user is creating. This way the user can see how it will look on the home page.

When searching for an artist the frontend will use the Musicbrainz API. This is explained in detail in section 2.5.
Users can only edit events for which they have permission to edit. Only the creator of an event has this permission.

2.3.2 List and view events

The events are listed on the homepage. When the user creates an account, he is required to choose 2 of his favourite genres. These genres are used to choose which events are shown first on the homepage. The hompage can be seen on figure 6.



Events this week you may like

Bart Peeters
Bonnefooi
Folk Alternative
0 December 22, 2023 19:00
20 EUR

Ariana Grande
Cirque Royal
Pop Singer-songwriter
0 December 22, 2023 19:00
65 EUR

Ninho
Beursschouwburg
Hiphop/Rap Afropop
0 December 22, 2023 19:00
35 EUR

Tonnie The man!
ING Arena
Rock Alternative
0 December 22, 2023 19:00
45 EUR

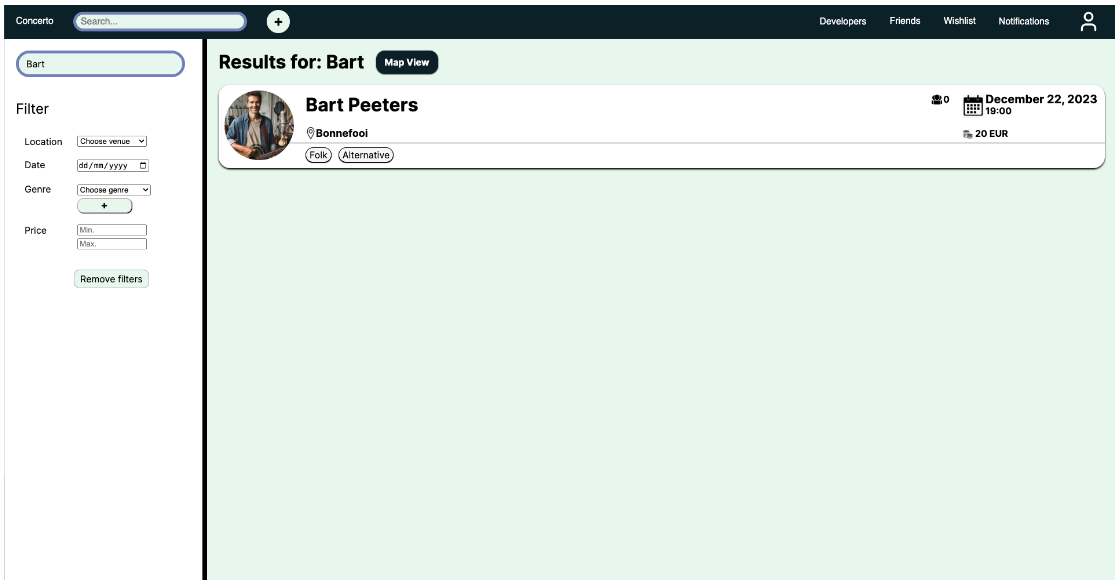
PLK en concert
Flagey Arts Center: Studio 4
Hiphop/Rap Alternative
0 December 22, 2023 19:15
10 EUR

0 December 22, 2023

Figure 6: The home page

2.3.3 Searching for information

On the homepage, the user can search for events by using their name. We also implemented a search bar in the navbar and on the friends page. The one on the navbar can search for users and events. The specific one on the friends page can only search in your friends. Figure 7 shows a search result on the homepage.



Results for: Bart

Bart Peeters
Bonnefooi
Folk Alternative
0 December 22, 2023 19:00
20 EUR

Figure 7: Search example on the homepage

We also implemented a filter system on the homepage. This filter system is used to filter the events listed on the homepage. A user can filter on location, the date, multiple genres, and the price range of an event. A user can also remove chosen filters.

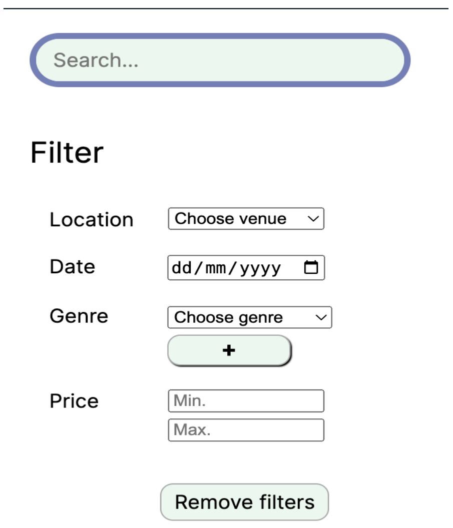
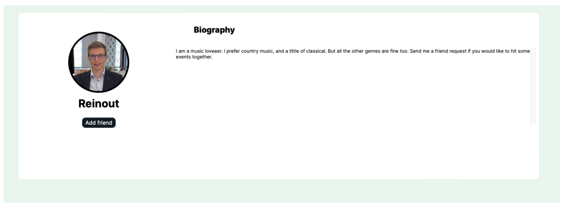


Figure 8: The filters for events

2.4 Social aspect

2.4.1 Friends

Users are able to send friend requests to other users. This is done by visiting their account. The user the request is sent to will get a notification. An example can be seen in figure 12. This notification will contain an accept and reject button. The sender will get a request when the receiver accepts the friend request. When the request hasn't been responded, the sender is able to undo the request and the notification at the receiver will be removed.



Biography

I am a music loveeer. I prefer country music, and a little of classical. But all the other genres are fine too. Send me a friend request if you would like to hit some events together.

Figure 9: Friend request button

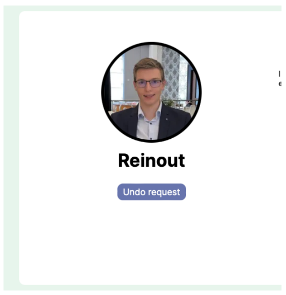


Figure 10: Undo friend request

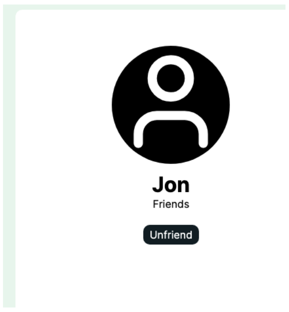
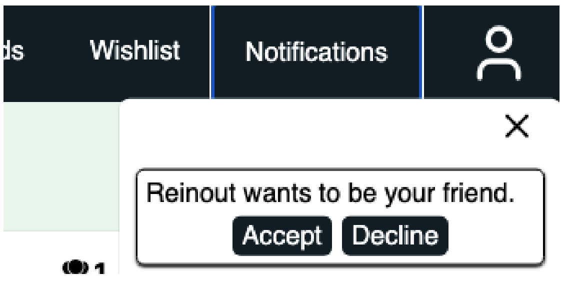


Figure 11: Removing a friend



Reinout wants to be your friend.

Figure 12: Friend request notification

When a user removes a friend no notifications are sent.

2.4.2 Event invites

When visiting an upcoming event the user is able to view all his friends which can be invited for that event. Friends which can be invited are not checked-in for that event and have not yet received an invite. When invited the receiver will get a notification which leads to the event when clicked. This can be seen in figure 13.

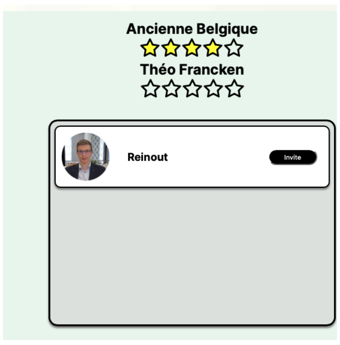


Figure 13: Friends which can be invited for an event

2.4.3 Reviews

When an event has finished all users which were checked-in will receive a notification to review both the artist and venue of that event. When clicking on this notification the user is redirected to the rating page. The user can only submit when both the artist and venue received a review. When viewing an event a user can click on the rating of the artist/venue to see a detailed overview where reviews from other users are shown. A review always has a score ranging from 1 - 5 and possibly a comment.

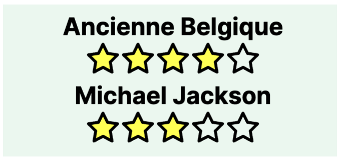
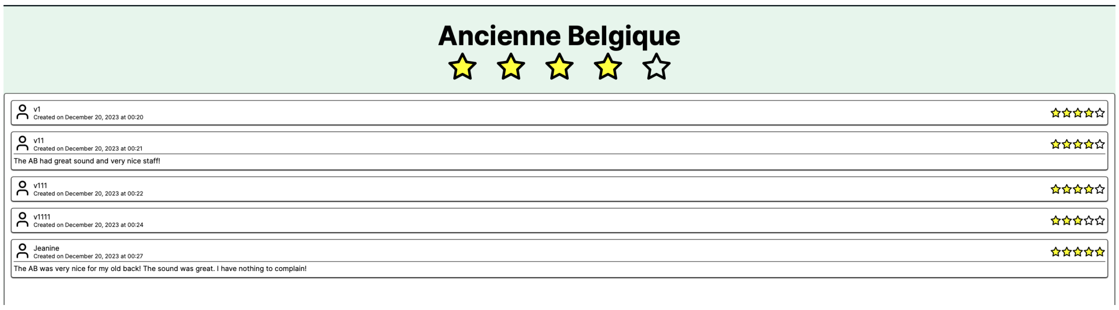


Figure 14: Reviews



v1
Created on December 20, 2023 at 00:20

v11
Created on December 20, 2023 at 00:21
The AB had great sound and very nice staff!

v111
Created on December 20, 2023 at 00:22

v1111
Created on December 20, 2023 at 00:24

Jeanine
Created on December 20, 2023 at 00:27
The AB was very nice for my old back! The sound was great. I have nothing to complain!

Figure 15: Reviews of a venue

2.5 Musicbrainz API

This API is utilized in different ways through our project. The back-end loads all the venues located in Brussels at startup. When an artist is mentioned in a request, the back-end first checks if the artist has been stored in the local database. If this is not the case it will attempt to fetch it from the API and store it locally.

The front-end only uses the API for artists. When a user searches an artist for an event it will perform a search on all the artists in the Musicbrainz database. This improves performance on the back-end as all the artist fetches when searching are performed client-sided.

There is a limit on the amount of requests we can send to the API. We are only allowed to send one request each second. Sending more requests than allowed will disable responses from the API for a while. This is an issue when multiple users are creating an event at the same time. When the backend receives multiple events with unknown artists, it needs to make a fetch for each artist. We thought of solving this issue by implementing a queue. Every second we would dequeue an artist and fetch its information. This was not possible because of 2 reasons:

1. If for example 50 users would create an event at the same time, the last user would have to wait 50 seconds before receiving a response, resulting in a time-out.
2. Musicbrainz does not allow scheduling on a free plan. When we tried to fetch every second they detected it and blocked us for a short amount of time.

This issue remains unsolved. When multiple of these requests are sent to the backend at the same time only one is executed, the others receive an error response.

2.6 Map view

A map is used at 2 places on our website. One on the homepage and one on each event page. By selecting map view on the home page you are presented with a map showing all the venues where an event takes place. Each marker represents a venue which you can select to view events hosted there. Clicking on these events redirects to the event page. You are also able to filter these events.

The second usage of a map is on an event's page. It shows a marker of the venue where it takes place, and shows a route from your location to the event.

For the map on the homepage we use Stadia Maps .

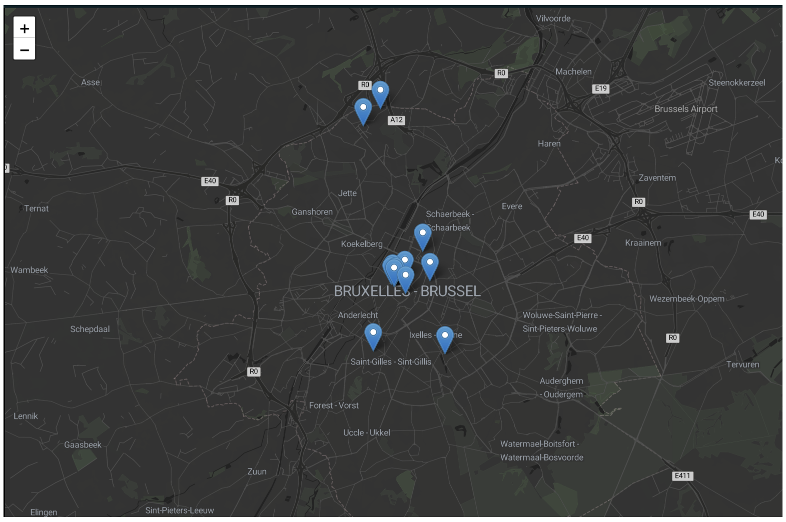


Figure 16: The homepage map

For the one on an event page we use Mapbox.

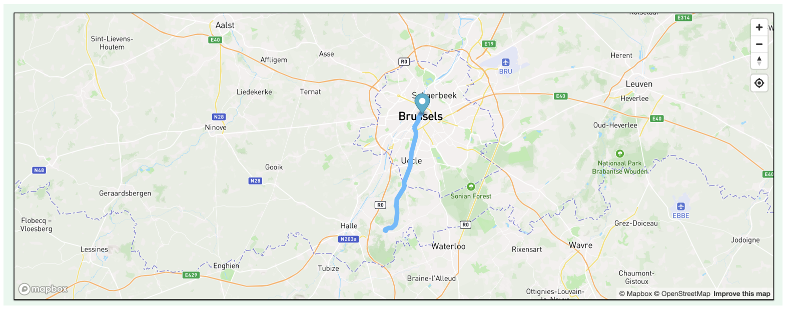


Figure 17: The event map page

2.7 Responsiveness

Our website is designed both for desktop as for a mobile device. To have this functionality we used media queries. This way we could switch between desktop and mobile version based on the screen width of the used device. Besides the layout that we needed to change, their was also a different approach needed for the navbar, sidebar on some pages and the drop down menu for the account. The navbar was changed into a hamburger menu. This hamburger menu is a drop down menu equipped with the same functionalities as the navbar.

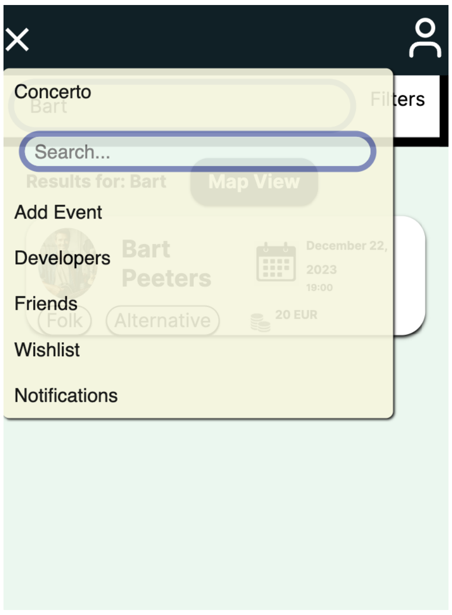


Figure 18: Hamburger menu

The sidebar was way too big to be show on a mobile device. It has been changed into a top bar that is displayed right under the navbar. Because of this decision, we couldn't shown all the filter functionalities on the homepage. This is why we changed the filter settings to a drop down menu.

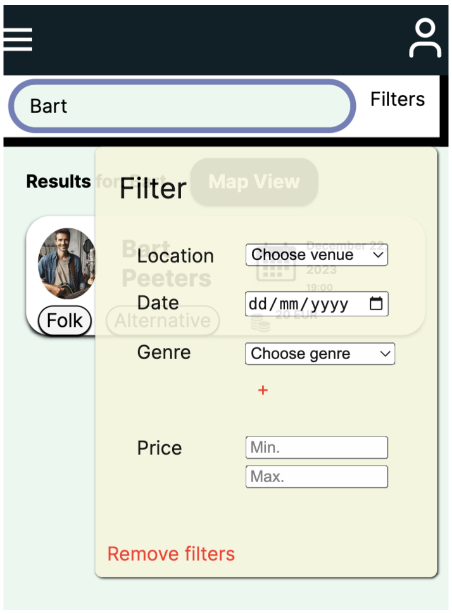


Figure 19: Drop-down for filter functionalities

When you're using a mobile device, you don't have the ability to hover over something. This is why the hover drop-down menu for the account had to change. When you click on the account picture in the top right corner on a mobile device, a drop-down menu is shown. This drop down menu contains all the functionalities regarding the account of a user.

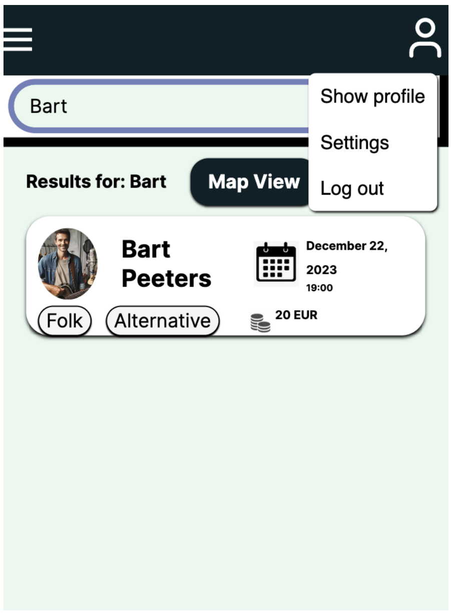


Figure 20: Drop-down for account functionalities

3 front-end

This section describes specific implementation details of the front-end.

3.1 Graphic design

Throughout the whole site we went with a minimalistic style. The site has a dark blue and light green theme. We tried to make the site as clear as possible by not displaying too much information on one page.

3.1.1 Content

We made sure that the website application used different images to give the website a more pleasant look. This also gives the user the ability to express them self. For example: making a custom profile picture.
On the homepage every listed event has its own picture. This picture is chosen by the creator of that specific event. These pictures add something extra to the look of the homepage. So the users have an influence over the looks of the homepage.

3.2 User interface

3.2.1 Navigation

We tried to keep the navigation on our web application as simple as possible. The user will mainly navigate through our website using the navbar. We added the most important pages to the navbar.

3.2.2 Interactive elements

The website has different interactive elements such as buttons, drop-down menu's, date pickers, etc... . They enhance the user's engagement and improve user experience. This makes the interface of our website more interactive and responsive.

3.2.3 Feedback

The user gets different feedback when using our site. For example when creating an event and not filling in all required fields, the user will get a notification message to fill in all the required fields.

3.2.4 Navbar

The navbar is a big part of our application. It allows the users to navigate to the desired page. Because of this, it is also the largest component based on lines of code. It offers redirecting to the login page when the user needs to be logged in to view that specific page. It also offers a search bar that allows the user to search events and users from the whole database, ignoring events that have already passed so that users can still view them. Notifications are also handled here.

3.2.5 Notifications

Notifications are a big part of the user experience. A user is notified when an event has finished to review the artist and venue. Since reviewing is big a part of our idea, notifications are a must. Users are also notified when they receive friend requests and when they are invited to an event by their friends. The user can then click on the notification to go to the concert page. When not checked in, the notification will stay until the user is checked in or the event has passed. Information notifications such as to notify the user that a friend request has been accepted will stay until the user has viewed the notification. After that, the notification is automatically removed.

3.2.6 Home and map page

The home page shows concerts that are happening this week. When the user is not logged in, these will be shown random. When the user is logged in, these will first be concerts that have one or more genres that the user likes. After that, random concerts will be shown. It only shows concerts for the current week. Monday would show events from Monday until Sunday. Sunday would only show events from Sunday. The map page shows all the venues where there are events this week. The user can click on a venue marker and see a list of events there.

3.2.7 Filtering

The home page offers filtering of events. Events can be filtered based on the venue, minimum and maximum price, date and genres. Genres can be added infinitely and removed one by one. Standard filtering is done based on events from the current week which are not expired, in the wish list or checked in. There is a search bar too. From the moment the search bar is used, the user can search for all events, including past events, wishlisted and checked in events. If there were already filters chosen, they will be applied on the search. This works the same on the map page.

3.2.8 Adding an event

The user can add an event by clicking on the *add-event button* in the navbar. On the add-event page, the user needs to fill in all the information of the event: title, banner image, event picture, description, date, time of doors opening, an optional time of the support act, time of the act itself, 2 genres, the location, the artist, the price and a URL to the page were tickets can be bought. All these fields are necessary and have client-side validation to check if the given information is correct. On submit, the data is also validated server-side.

3.3 Implementation

Our website uses the Next.js framework. This is a React framework that offers great support for server-side rendering.

3.3.1 Libraries

The only libraries that were used in the front-end were:

- *mapbox and react-map-gl* to display the location of an event on the event page. It uses the Routing API of Mapbox.
- *react-leaflet* for Stadia maps to display the map with all the venues.

- *lodash* for one line in our code to see if two objects are equal.

3.3.2 Javascript

In our project, we used typescript. Our implementation utilizes *useState* variables. These are variables that can change dynamically. Without them, the page would be static as normal constants and variables wouldn't change during execution. These dynamic variables offer the possibility of conditional rendering. *useEffect* hooks are also widely used in our code. They use a dependency array that tells the useEffect hook to execute when a value in the dependency array changes. *useCallback* hooks are used as well to "memoize" some functions.

3.3.3 HTML

We used a lot of *div* elements because they do not have standard styling, which made it easier when adding CSS styling. We also used some of the HTML5 elements like *input type, date, select, time, ...*

3.3.4 CSS

The layout for the front-end is written in CSS. We wrote all the CSS ourselves. We structured the CSS code in logical CSS files. All the pages can be found in *Home.module.css.* All the components have their own CSS file. No utility classes were used. No TailWind CSS was used.

4 back-end

This section describes specific implementation details of the back-end.

4.1 Frameworks and middleware

For the back-end side of our application Node.js and Express were used. Other middleware which we used:

- Bcrypt was used for password security. This middleware hashes and compares passwords.
- Express-validator is used to perform validation on all routes.
- Node-cron provides functionality to perform a function repeatedly at a certain time or interval. For our application it was used to notify checked-in users of finished events to leave a review. For now this procedure is executed every minute. This allowed easier testing and filling of the database. In a real world setting this can be done every 24 hours to prevent redundant checks in the database.
- Express-session is used to implement sessions.
- Sequelize in combination with Sqlite3 to implement the database.
- Nodemailer is used to sent an email to a user upon registration.

4.2 Relations between data

The database in the backend contains a lot of tables which have relations between each other. This section is dedicated to describing each table and its relations. In appendix A all tables can be found. Each instance always has an ID used to identify it in the application.

Figure 21 shows the user class from the database. All attributes are required fields. Most fields are self-explanatory but I will explain the privacy fields. Each user has 3 settings related to privacy, each has one of the following three values: public, friends or private. In our implementation these are individually stored for each users. A different implementation could be to store all possible combinations in a different table, each with a unique ID, and then store the ID corresponding to the user's settings. We thought of implementing this to prevent redundant information. However we did not think it was necessary because the data was very limited for this project. However this can be implemented in the future.

Figure 22 shows how events are stored within the database. All fields are mandatory except the "support" attribute. This is a string "HH:MM" indicating when the supporting artist starts, but as not all events have a support artist this is optional.

Figure 23 displays how check-ins from users are stored in the database. This is a many-to-many relationship between users and events. As multiple users can check-in for an event, and one user can check-in for multiple events. The same can be said about wishlisting events which can be seen in figure 24.

How artists are stored can be seen in 25. The type of an artist describes if it is a "band", an "orchestre", etc... Venues are stored in figure 26. Each venue stores its coordinates. This enables the frontend to display them correctly on maps.

Users can also become friends with each other. This can be seen in figure 27. Each friendship is identified by an ID and a status. The status is either "accepted" or "pending". When a friend request is sent from one user to another a friendship instance is created with status "pending". This table represents a many-to-many relationship between the user table itself.

Each artist and venue have a score based on the reviews they has received. One rating object is stored for each, so they have a one-to-one relationship. Reviews belong to rating objects and contain the score a user has given and an optional message. This is a one-to-many relationship between reviews and rating objects, as each review belongs to one rating object and a rating object can have multiple reviews.

Notifications from figure 31 are implemented in a similar manner. There is a one-to-many relationship between notifications and users, as each notification belongs to one user while a user can have multiple notifications. The status attribute indicates if the notification has been read or not. Additionally each notificationobject from figure 30 has a one-to-many relationship with a notification. This has been done because the same notification might be sent to multiple users. Each user then has its own notification which can be deleted and marked as seen, without influencing other similar notifications. So the information about a notification is stored in a notificationObject, and the instance interacting with the user in a notification. The "notificationType" can be: "friendRequestReceived", "reviewEvent", etc... The actor is the reason why the notification was sent. This can be null (if it was send by the web application itself) or a number (the userID of the user which has triggered the notification).

4.3 Security

All information is directly stored in the database, except the password. It is first salted + hashed before being stored. Passwords never leave the back-end side of our application. If they would be leaked, the hashed password cannot be used to login. For each user a unique ID is generated. It is used for all user operations.

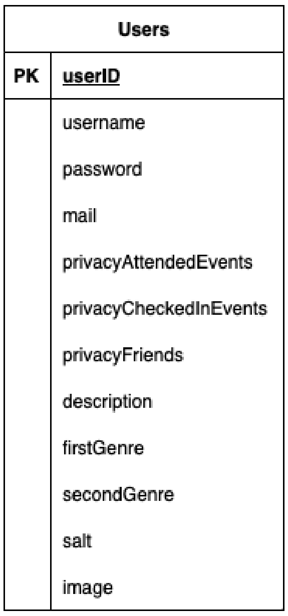
Authentication is implemented by using session-cookies. Every response from the back-end contains a session-cookie, that the front-end can provide with each request. These sessions expire after 15 minutes. When logged-in the user-ID from that user is stored in the session. This implementation allows the back-end to check if requests are authorized. Personalized information retrieval is also possible this way: a GET request to“/profile” returns information about the logged-in user.

4.4 Notifications

Earlier in the project we implemented server-sent events, where the frontend could subscribe to notifications by sending a GET request to /notifications/subscribe . However, after a while we discovered several bugs with connections not closing properly, and notifications being broadcasted to all subscribed users instead of only one. For this reason we decided to remove this feature.

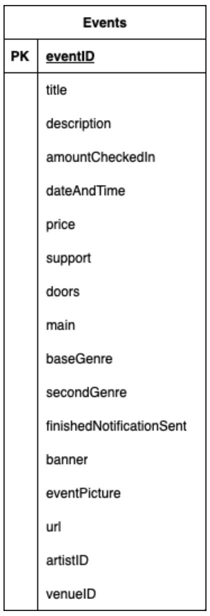
In the final version of our project users do not receive live notifications. They have to refresh the page to receive new ones. Implementing live notifications could be possible in the future.

A Database tables



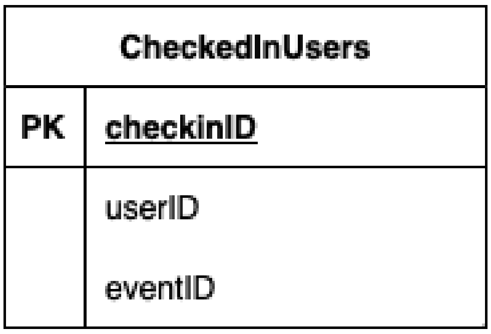
Users	
PK	userID
	username
	password
	mail
	privacyAttendedEvents
	privacyCheckedInEvents
	privacyFriends
	description
	firstGenre
	secondGenre
	salt
	image

Figure 21: the table stored for users



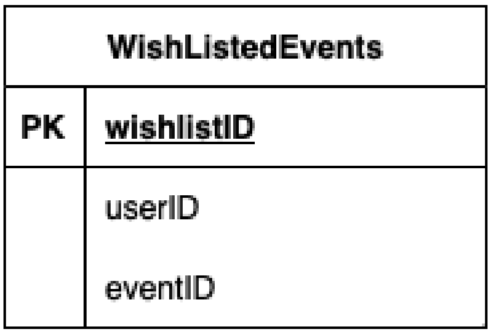
Events	
PK	eventID
	title
	description
	amountCheckedIn
	dateAndTime
	price
	support
	doors
	main
	baseGenre
	secondGenre
	finishedNotificationSent
	banner
	eventPicture
	url
	artistID
	venueID

Figure 22: the table stored for events



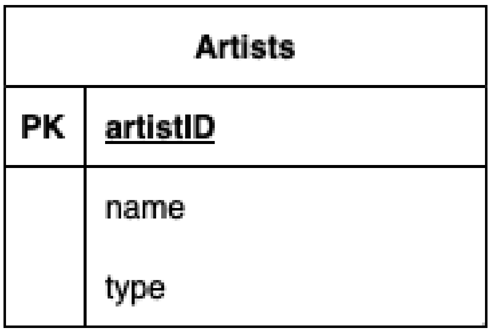
CheckedInUsers	
PK	checkinID
	userID
	eventID

Figure 23: Table which stores users checking-in to events



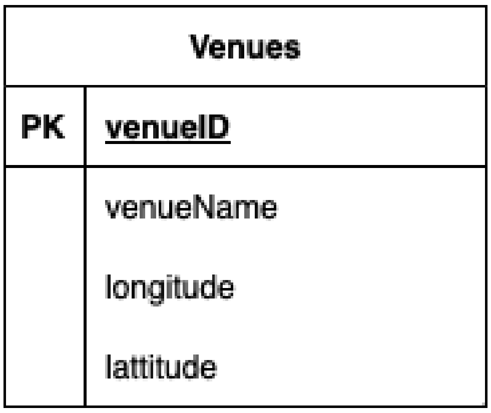
WishListedEvents	
PK	wishlistID
	userID
	eventID

Figure 24: Table to store events wishlisted by users



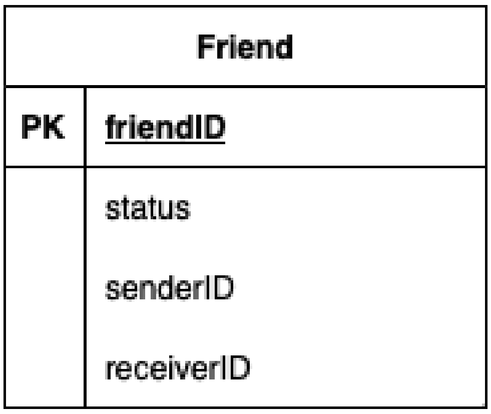
Artists	
PK	artistID
	name
	type

Figure 25: The table which stores artists



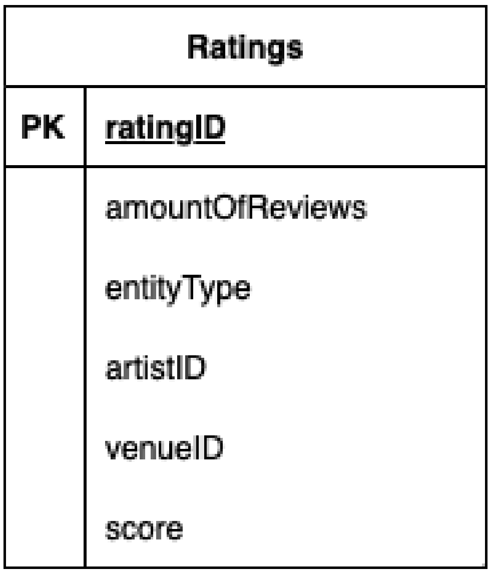
Venues	
PK	venueID
	venueName
	longitude
	lattitude

Figure 26: Table to store venues



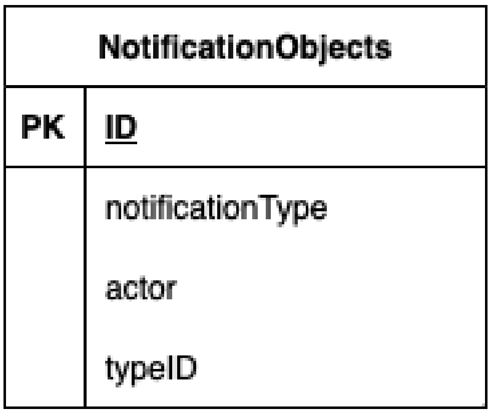
	Friend
PK	friendID
	status
	senderID
	receiverID

Figure 27: Table to store information related to friend relationship between users



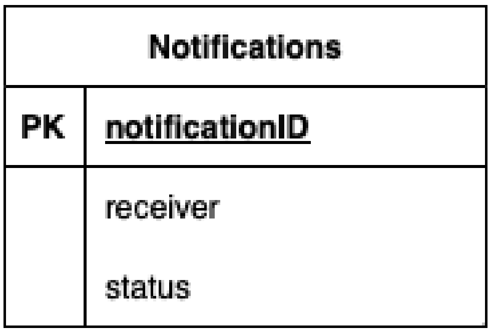
	Ratings
PK	ratingID
	amountOfReviews
	entityType
	artistID
	venueID
	score

Figure 28: Table to store the rating for an event



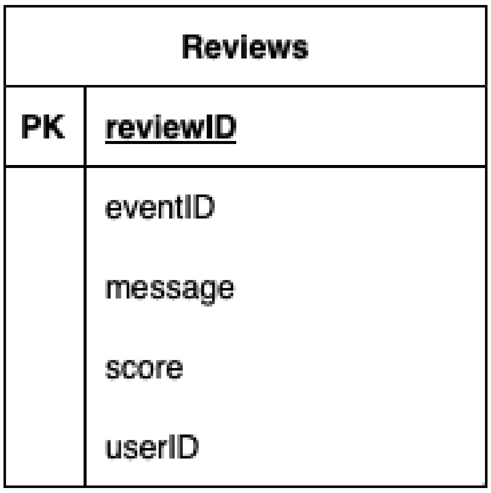
NotificationObjects	
PK	<u>ID</u>
	notificationType
	actor
	typeID

Figure 30: Table to store a notification object



Notifications	
PK	<u>notificationID</u>
	receiver
	status

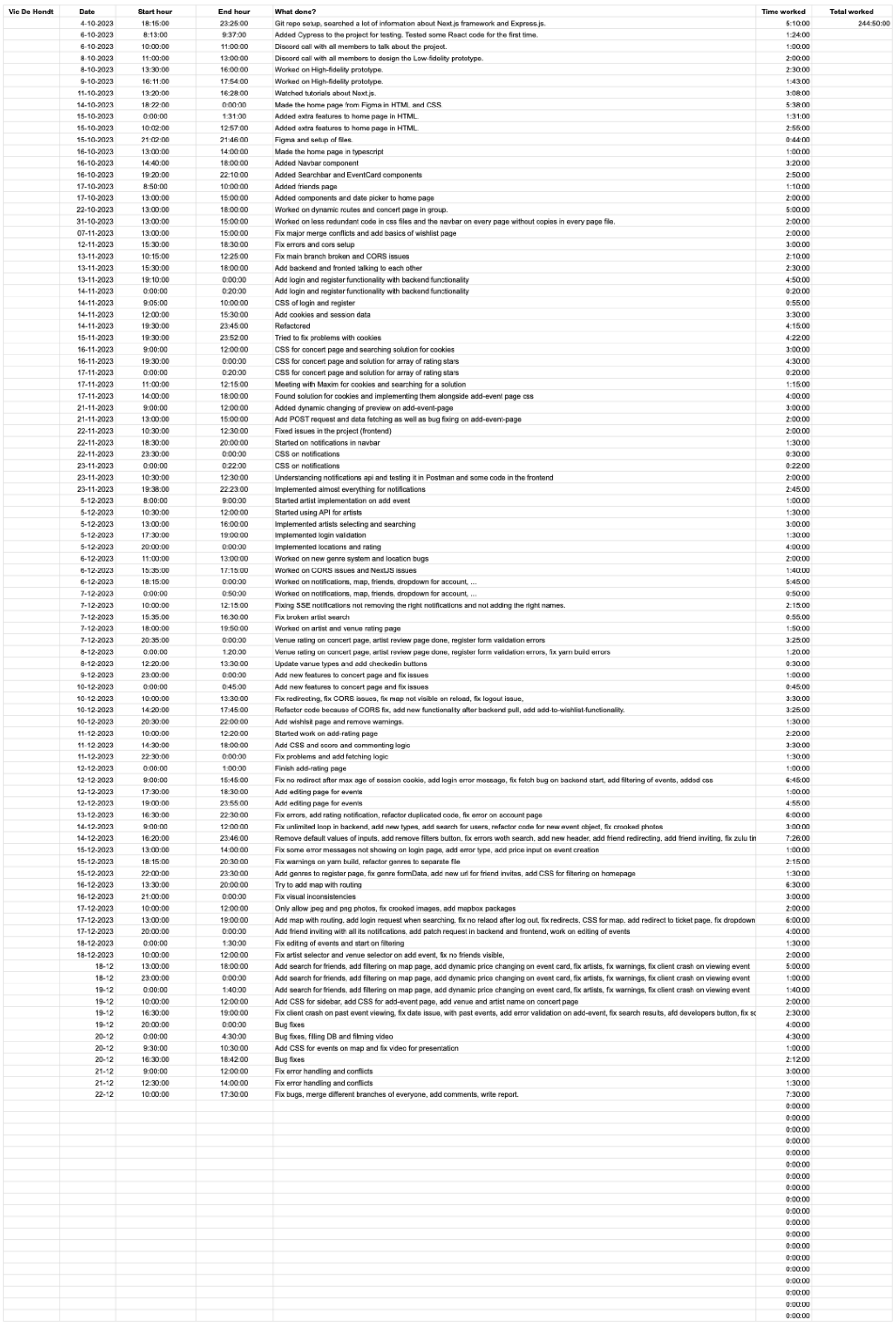
Figure 31: Table to store notifications



Reviews	
PK	<u>reviewID</u>
	eventID
	message
	score
	userID

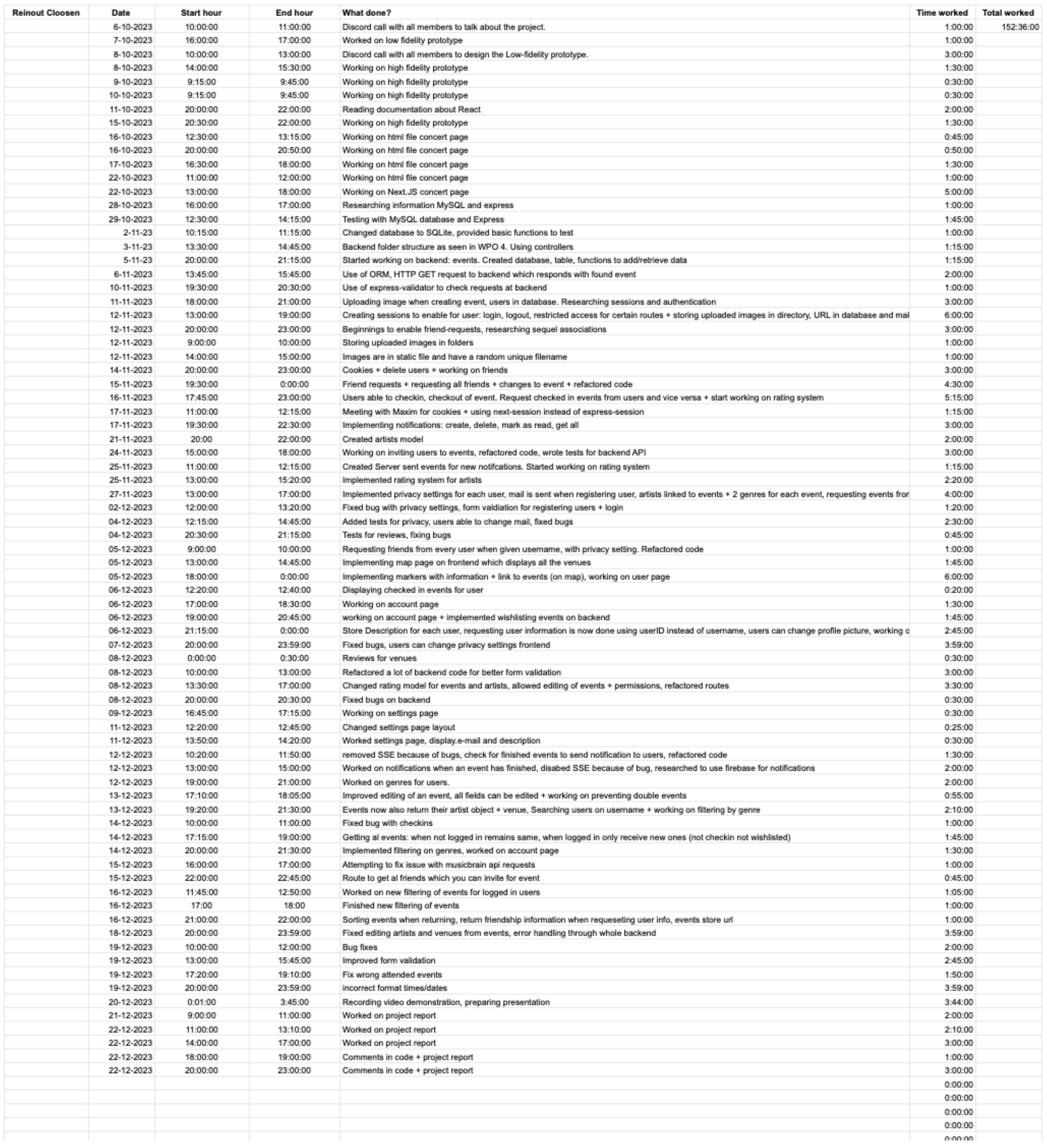
Figure 29: Table to store user reviews

B Timesheets



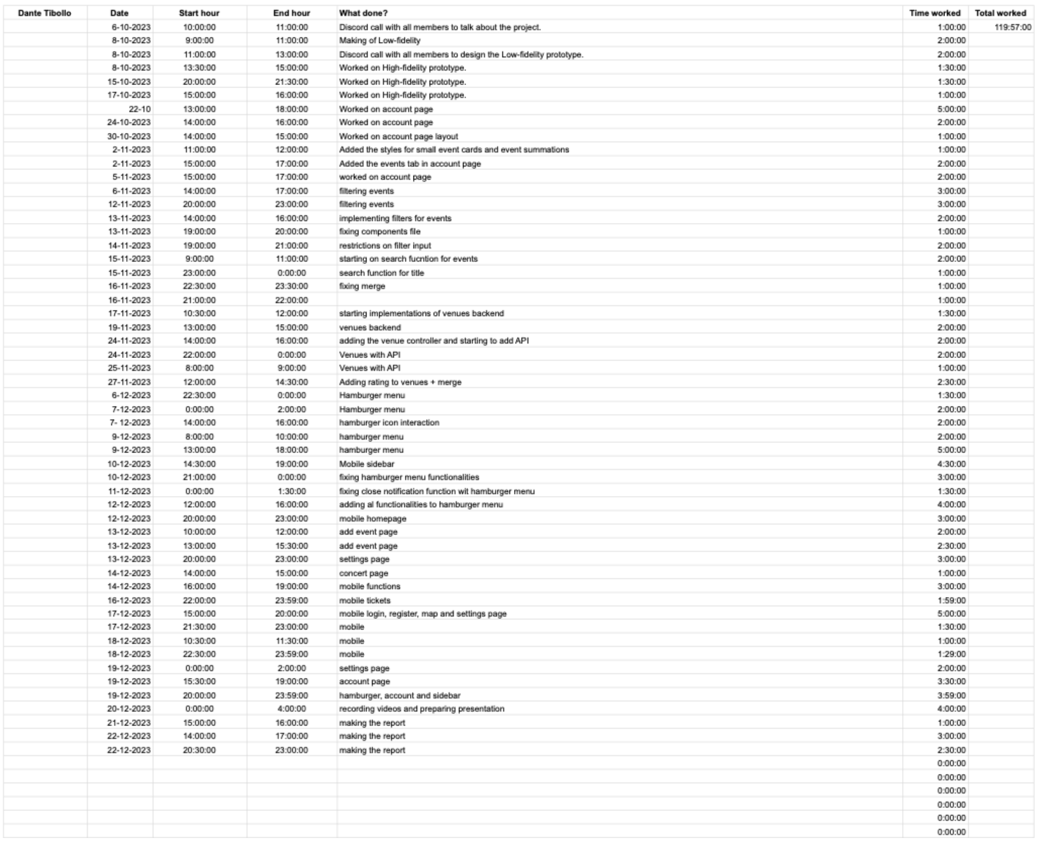
Vic De Hondt	Date	Start hour	End hour	What done?	Time worked	Total worked
	4-10-2023	18:15:00	23:25:00	Git repo setup, searched a lot of information about Next.js framework and Express.js.	5:10:00	244:50:00
	6-10-2023	8:13:00	9:37:00	Added Cypress to the project for testing. Tested some React code for the first time.	1:24:00	
	6-10-2023	10:00:00	11:00:00	Discord call with all members to talk about the project.	1:00:00	
	8-10-2023	11:00:00	13:00:00	Discord call with all members to design the Low-fidelity prototype.	2:00:00	
	8-10-2023	13:30:00	16:00:00	Worked on High-fidelity prototype.	2:30:00	
	9-10-2023	16:11:00	17:54:00	Worked on High-fidelity prototype.	1:43:00	
	11-10-2023	13:20:00	16:28:00	Watched tutorials about Next.js.	3:08:00	
	14-10-2023	18:22:00	0:00:00	Made the home page from Figma in HTML and CSS.	5:38:00	
	15-10-2023	0:00:00	1:31:00	Added extra features to home page in HTML.	1:31:00	
	15-10-2023	10:02:00	12:57:00	Added extra features to home page in HTML.	2:55:00	
	15-10-2023	21:02:00	21:46:00	Figma and setup of files.	0:44:00	
	16-10-2023	13:00:00	14:00:00	Made the home page in typescript	1:00:00	
	16-10-2023	14:40:00	18:00:00	Added Navbar component	3:20:00	
	16-10-2023	19:20:00	22:10:00	Added Searchbar and EventCard components	2:50:00	
	17-10-2023	8:50:00	10:00:00	Added friends page	1:10:00	
	17-10-2023	13:00:00	15:00:00	Added components and date picker to home page	2:00:00	
	22-10-2023	13:00:00	18:00:00	Worked on dynamic routes and concert page in group.	5:00:00	
	31-10-2023	13:00:00	15:00:00	Worked on less redundant code in css files and the navbar on every page without copies in every page file.	2:00:00	
	07-11-2023	13:00:00	15:00:00	Fix major merge conflicts and add basics of wishlist page	2:00:00	
	12-11-2023	15:30:00	18:30:00	Fix errors and cors setup	3:00:00	
	13-11-2023	10:15:00	12:25:00	Fix main branch broken and CORS issues	2:10:00	
	13-11-2023	15:30:00	18:00:00	Add backend and fronted talking to each other	2:30:00	
	13-11-2023	19:10:00	0:00:00	Add login and register functionality with backend functionality	4:50:00	
	14-11-2023	0:00:00	0:20:00	Add login and register functionality with backend functionality	0:20:00	
	14-11-2023	9:05:00	10:00:00	CSS of login and register	0:55:00	
	14-11-2023	12:00:00	15:30:00	Add cookies and session data	3:30:00	
	14-11-2023	19:30:00	23:45:00	Refactored	4:15:00	
	15-11-2023	19:30:00	23:52:00	Tried to fix problems with cookies	4:22:00	
	16-11-2023	9:00:00	12:00:00	CSS for concert page and searching solution for cookies	3:00:00	
	16-11-2023	19:30:00	0:00:00	CSS for concert page and solution for array of rating stars	4:30:00	
	17-11-2023	0:00:00	0:20:00	CSS for concert page and solution for array of rating stars	0:20:00	
	17-11-2023	11:00:00	12:15:00	Meeting with Maxim for cookies and searching for a solution	1:15:00	
	17-11-2023	14:00:00	18:00:00	Found solution for cookies and implementing them alongside add-event page css	4:00:00	
	21-11-2023	9:00:00	12:00:00	Added dynamic changing of preview on add-event-page	3:00:00	
	21-11-2023	13:00:00	15:00:00	Add POST request and data fetching as well as bug fixing on add-event-page	2:00:00	
	22-11-2023	10:30:00	12:30:00	Fixed issues in the project (frontend)	2:00:00	
	22-11-2023	18:30:00	20:00:00	Started on notifications in navbar	1:30:00	
	22-11-2023	23:30:00	0:00:00	CSS on notifications	0:30:00	
	23-11-2023	0:00:00	0:22:00	CSS on notifications	0:22:00	
	23-11-2023	10:30:00	12:30:00	Understanding notifications api and testing it in Postman and some code in the frontend	2:00:00	
	23-11-2023	19:38:00	22:23:00	Implemented almost everything for notifications	2:45:00	
	5-12-2023	8:00:00	9:00:00	Started artist implementation on add event	1:00:00	
	5-12-2023	10:30:00	12:00:00	Started using API for artists	1:30:00	
	5-12-2023	13:00:00	16:00:00	Implemented artists selecting and searching	3:00:00	
	5-12-2023	17:30:00	19:00:00	Implemented login validation	1:30:00	
	5-12-2023	20:00:00	0:00:00	Implemented locations and rating	4:00:00	
	6-12-2023	11:00:00	13:00:00	Worked on new genre system and location bugs	2:00:00	
	6-12-2023	15:35:00	17:15:00	Worked on CORS issues and NextJS issues	1:40:00	
	6-12-2023	18:15:00	0:00:00	Worked on notifications, map, friends, dropdown for account, ...	5:45:00	
	7-12-2023	0:00:00	0:50:00	Worked on notifications, map, friends, dropdown for account, ...	0:50:00	
	7-12-2023	10:00:00	12:15:00	Fixing SSE notifications not removing the right notifications and not adding the right names.	2:15:00	
	7-12-2023	15:35:00	16:30:00	Fix broken artist search	0:55:00	
	7-12-2023	18:00:00	19:50:00	Worked on artist and venue rating page	1:50:00	
	7-12-2023	20:35:00	0:00:00	Venue rating on concert page, artist review page done, register form validation errors	3:25:00	
	8-12-2023	0:00:00	1:20:00	Venue rating on concert page, artist review page done, register form validation errors, fix yarn build errors	1:20:00	
	8-12-2023	12:20:00	13:30:00	Update vanue types and add checkedin buttons	0:30:00	
	9-12-2023	23:00:00	0:00:00	Add new features to concert page and fix issues	1:00:00	
	10-12-2023	0:00:00	0:45:00	Add new features to concert page and fix issues	0:45:00	
	10-12-2023	10:00:00	13:30:00	Fix redirecting, fix CORS issues, fix map not visible on reload, fix logout issue,	3:30:00	
	10-12-2023	14:20:00	17:45:00	Refactor code because of CORS fix, add new functionality after backend pull, add add-to-wishlist-functionality.	3:25:00	
	10-12-2023	20:30:00	22:00:00	Add wishlsit page and remove warnings.	1:30:00	
	11-12-2023	10:00:00	12:20:00	Started work on add-rating page	2:20:00	
	11-12-2023	14:30:00	18:00:00	Add CSS and score and commenting logic	3:30:00	
	11-12-2023	22:30:00	0:00:00	Fix problems and add fetching logic	1:30:00	
	12-12-2023	0:00:00	1:00:00	Finish add-rating page	1:00:00	
	12-12-2023	9:00:00	15:45:00	Fix no redirect after max age of session cookie, add login error message, fix fetch bug on backend start, add filtering of events, added css	6:45:00	
	12-12-2023	17:30:00	18:30:00	Add editing page for events	1:00:00	
	12-12-2023	19:00:00	23:55:00	Add editing page for events	4:55:00	
	13-12-2023	16:30:00	22:30:00	Fix errors, add rating notification, refactor duplicated code, fix error on account page	6:00:00	
	14-12-2023	9:00:00	12:00:00	Fix unlimited loop in backend, add new types, add search for users, refactor code for new event object, fix crooked photos	3:00:00	
	14-12-2023	16:20:00	23:46:00	Remove default values of inputs, add remove filters button, fix errors woth search, add new header, add friend redirecting, add friend inviting, fix zulu tin	7:26:00	
	15-12-2023	13:00:00	14:00:00	Fix some error messages not showing on login page, add error type, add price input on event creation	1:00:00	
	15-12-2023	18:15:00	20:30:00	Fix warnings on yarn build, refactor genres to separate file	2:15:00	
	15-12-2023	22:00:00	23:30:00	Add genres to register page, fix genre formData, add new url for friend invites, add CSS for filtering on homepage	1:30:00	
	16-12-2023	13:30:00	20:00:00	Try to add map with routing	6:30:00	
	16-12-2023	21:00:00	0:00:00	Fix visual inconsistencies	3:00:00	
	17-12-2023	10:00:00	12:00:00	Only allow jpeg and png photos, fix crooked images, add mapbox packages	2:00:00	
	17-12-2023	13:00:00	19:00:00	Add map with routing, add login request when searching, fix no relaod after log out, fix redirects, CSS for map, add redirect to ticket page, fix dropdown	6:00:00	
	17-12-2023	20:00:00	0:00:00	Add friend inviting with all its notifications, add patch request in backend and frontend, work on editing of events	4:00:00	
	18-12-2023	0:00:00	1:30:00	Fix editing of events and start on filtering	1:30:00	
	18-12-2023	10:00:00	12:00:00	Fix artist selector and venue selector on add event, fix no friends visible,	2:00:00	
	18-12	13:00:00	18:00:00	Add search for friends, add filtering on map page, add dynamic price changing on event card, fix artists, fix warnings, fix client crash on viewing event	5:00:00	
	18-12	23:00:00	0:00:00	Add search for friends, add filtering on map page, add dynamic price changing on event card, fix artists, fix warnings, fix client crash on viewing event	1:00:00	
	19-12	0:00:00	1:40:00	Add search for friends, add filtering on map page, add dynamic price changing on event card, fix artists, fix warnings, fix client crash on viewing event	1:40:00	
	19-12	10:00:00	12:00:00	Add CSS for sidebar, add CSS for add-event page, add venue and artist name on concert page	2:00:00	
	19-12	16:30:00	19:00:00	Fix client crash on past event viewing, fix date issue, with past events, add error validation on add-event, fix search results, afd developers button, fix sc	2:30:00	
	19-12	20:00:00	0:00:00	Bug fixes	4:00:00	
	20-12	0:00:00	4:30:00	Bug fixes, filling DB and filming video	4:30:00	
	20-12	9:30:00	10:30:00	Add CSS for events on map and fix video for presentation	1:00:00	
	20-12	16:30:00	18:42:00	Bug fixes	2:12:00	
	21-12	9:00:00	12:00:00	Fix error handling and conflicts	3:00:00	
	21-12	12:30:00	14:00:00	Fix error handling and conflicts	1:30:00	
	22-12	10:00:00	17:30:00	Fix bugs, merge different branches of everyone, add comments, write report.	7:30:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	

Figure 32: Vic his timesheet



Reinout Cloosen	Date	Start hour	End hour	What done?	Time worked	Total worked
	6-10-2023	10:00:00	11:00:00	Discord call with all members to talk about the project.	1:00:00	152:36:00
	7-10-2023	16:00:00	17:00:00	Worked on low fidelity prototype	1:00:00	
	8-10-2023	10:00:00	13:00:00	Discord call with all members to design the Low-fidelity prototype.	3:00:00	
	8-10-2023	14:00:00	15:30:00	Working on high fidelity prototype	1:30:00	
	9-10-2023	9:15:00	9:45:00	Working on high fidelity prototype	0:30:00	
	10-10-2023	9:15:00	9:45:00	Working on high fidelity prototype	0:30:00	
	11-10-2023	20:00:00	22:00:00	Reading documentation about React	2:00:00	
	15-10-2023	20:30:00	22:00:00	Working on high fidelity prototype	1:30:00	
	16-10-2023	12:30:00	13:15:00	Working on html file concert page	0:45:00	
	16-10-2023	20:00:00	20:50:00	Working on html file concert page	0:50:00	
	17-10-2023	16:30:00	18:00:00	Working on html file concert page	1:30:00	
	22-10-2023	11:00:00	12:00:00	Working on html file concert page	1:00:00	
	22-10-2023	13:00:00	18:00:00	Working on Next.JS concert page	5:00:00	
	28-10-2023	16:00:00	17:00:00	Researching information MySQL and express	1:00:00	
	29-10-2023	12:30:00	14:15:00	Testing with MySQL database and Express	1:45:00	
	2-11-23	10:15:00	11:15:00	Changed database to SQLite, provided basic functions to test	1:00:00	
	3-11-23	13:30:00	14:45:00	Backend folder structure as seen in WPO 4. Using controllers	1:15:00	
	5-11-23	20:00:00	21:15:00	Started working on backend: events. Created database, table, functions to add/retrieve data	1:15:00	
	6-11-2023	13:45:00	15:45:00	Use of ORM, HTTP GET request to backend which responds with found event	2:00:00	
	10-11-2023	19:30:00	20:30:00	Use of express-validator to check requests at backend	1:00:00	
	11-11-2023	18:00:00	21:00:00	Uploading image when creating event, users in database. Researching sessions and authentication	3:00:00	
	12-11-2023	13:00:00	19:00:00	Creating sessions to enable for user: login, logout, restricted access for certain routes + storing uploaded images in directory, URL in database and mai	6:00:00	
	12-11-2023	20:00:00	23:00:00	Beginnings to enable friend-requests, researching sequel associations	3:00:00	
	12-11-2023	9:00:00	10:00:00	Storing uploaded images in folders	1:00:00	
	12-11-2023	14:00:00	15:00:00	Images are in static file and have a random unique filename	1:00:00	
	14-11-2023	20:00:00	23:00:00	Cookies + delete users + working on friends	3:00:00	
	15-11-2023	19:30:00	0:00:00	Friend requests + requesting all friends + changes to event + refactored code	4:30:00	
	16-11-2023	17:45:00	23:00:00	Users able to checkin, checkout of event. Request checked in events from users and vice versa + start working on rating system	5:15:00	
	17-11-2023	11:00:00	12:15:00	Meeting with Maxim for cookies + using next-session instead of express-session	1:15:00	
	17-11-2023	19:30:00	22:30:00	Implementing notifications: create, delete, mark as read, get all	3:00:00	
	21-11-2023	20:00	22:00:00	Created artists model	2:00:00	
	24-11-2023	15:00:00	18:00:00	Working on inviting users to events, refactored code, wrote tests for backend API	3:00:00	
	25-11-2023	11:00:00	12:15:00	Created Server sent events for new notifcations. Started working on rating system	1:15:00	
	25-11-2023	13:00:00	15:20:00	Implemented rating system for artists	2:20:00	
	27-11-2023	13:00:00	17:00:00	Implemented privacy settings for each user, mail is sent when registering user, artists linked to events + 2 genres for each event, requesting events fror	4:00:00	
	02-12-2023	12:00:00	13:20:00	Fixed bug with privacy settings, form valdiation for registering users + login	1:20:00	
	04-12-2023	12:15:00	14:45:00	Added tests for privacy, users able to change mail, fixed bugs	2:30:00	
	04-12-2023	20:30:00	21:15:00	Tests for reviews, fixing bugs	0:45:00	
	05-12-2023	9:00:00	10:00:00	Requesting friends from every user when given username, with privacy setting. Refactored code	1:00:00	
	05-12-2023	13:00:00	14:45:00	Implementing map page on frontend which displays all the venues	1:45:00	
	05-12-2023	18:00:00	0:00:00	Implementing markers with information + link to events (on map), working on user page	6:00:00	
	06-12-2023	12:20:00	12:40:00	Displaying checked in events for user	0:20:00	
	06-12-2023	17:00:00	18:30:00	Working on account page	1:30:00	
	06-12-2023	19:00:00	20:45:00	working on account page + implemented wishlisting events on backend	1:45:00	
	06-12-2023	21:15:00	0:00:00	Store Description for each user, requesting user information is now done using userID instead of username, users can change profile picture, working c	2:45:00	
	07-12-2023	20:00:00	23:59:00	Fixed bugs, users can change privacy settings frontend	3:59:00	
	08-12-2023	0:00:00	0:30:00	Reviews for venues	0:30:00	
	08-12-2023	10:00:00	13:00:00	Refactored a lot of backend code for better form validation	3:00:00	
	08-12-2023	13:30:00	17:00:00	Changed rating model for events and artists, allowed editing of events + permissions, refactored routes	3:30:00	
	08-12-2023	20:00:00	20:30:00	Fixed bugs on backend	0:30:00	
	09-12-2023	16:45:00	17:15:00	Working on settings page	0:30:00	
	11-12-2023	12:20:00	12:45:00	Changed settings page layout	0:25:00	
	11-12-2023	13:50:00	14:20:00	Worked settings page, display.e-mail and description	0:30:00	
	12-12-2023	10:20:00	11:50:00	removed SSE because of bugs, check for finished events to send notification to users, refactored code	1:30:00	
	12-12-2023	13:00:00	15:00:00	Worked on notifications when an event has finished, disabed SSE because of bug, researched to use firebase for notifications	2:00:00	
	12-12-2023	19:00:00	21:00:00	Worked on genres for users.	2:00:00	
	13-12-2023	17:10:00	18:05:00	Improved editing of an event, all fields can be edited + working on preventing double events	0:55:00	
	13-12-2023	19:20:00	21:30:00	Events now also return their artist object + venue, Searching users on username + working on filtering by genre	2:10:00	
	14-12-2023	10:00:00	11:00:00	Fixed bug with checkins	1:00:00	
	14-12-2023	17:15:00	19:00:00	Getting al events: when not logged in remains same, when logged in only receive new ones (not checkin not wishlisted)	1:45:00	
	14-12-2023	20:00:00	21:30:00	Implemented filtering on genres, worked on account page	1:30:00	
	15-12-2023	16:00:00	17:00:00	Attempting to fix issue with musicbrain api requests	1:00:00	
	15-12-2023	22:00:00	22:45:00	Route to get al friends which you can invite for event	0:45:00	
	16-12-2023	11:45:00	12:50:00	Worked on new filtering of events for logged in users	1:05:00	
	16-12-2023	17:00	18:00	Finished new filtering of events	1:00:00	
	16-12-2023	21:00:00	22:00:00	Sorting events when returning, return friendship information when requeseting user info, events store url	1:00:00	
	18-12-2023	20:00:00	23:59:00	Fixed editing artists and venues from events, error handling through whole backend	3:59:00	
	19-12-2023	10:00:00	12:00:00	Bug fixes	2:00:00	
	19-12-2023	13:00:00	15:45:00	Improved form validation	2:45:00	
	19-12-2023	17:20:00	19:10:00	Fix wrong attended events	1:50:00	
	19-12-2023	20:00:00	23:59:00	incorrect format times/dates	3:59:00	
	20-12-2023	0:01:00	3:45:00	Recording video demonstration, preparing presentation	3:44:00	
	21-12-2023	9:00:00	11:00:00	Worked on project report	2:00:00	
	22-12-2023	11:00:00	13:10:00	Worked on project report	2:10:00	
	22-12-2023	14:00:00	17:00:00	Worked on project report	3:00:00	
	22-12-2023	18:00:00	19:00:00	Comments in code + project report	1:00:00	
	22-12-2023	20:00:00	23:00:00	Comments in code + project report	3:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	

Figure 33: Reinout his timesheet



Dante Tibollo	Date	Start hour	End hour	What done?	Time worked	Total worked
	6-10-2023	10:00:00	11:00:00	Discord call with all members to talk about the project.	1:00:00	119:57:00
	8-10-2023	9:00:00	11:00:00	Making of Low-fidelity	2:00:00	
	8-10-2023	11:00:00	13:00:00	Discord call with all members to design the Low-fidelity prototype.	2:00:00	
	8-10-2023	13:30:00	15:00:00	Worked on High-fidelity prototype.	1:30:00	
	15-10-2023	20:00:00	21:30:00	Worked on High-fidelity prototype.	1:30:00	
	17-10-2023	15:00:00	16:00:00	Worked on High-fidelity prototype.	1:00:00	
	22-10	13:00:00	18:00:00	Worked on account page	5:00:00	
	24-10-2023	14:00:00	16:00:00	Worked on account page	2:00:00	
	30-10-2023	14:00:00	15:00:00	Worked on account page layout	1:00:00	
	2-11-2023	11:00:00	12:00:00	Added the styles for small event cards and event summations	1:00:00	
	2-11-2023	15:00:00	17:00:00	Added the events tab in account page	2:00:00	
	5-11-2023	15:00:00	17:00:00	worked on account page	2:00:00	
	6-11-2023	14:00:00	17:00:00	filtering events	3:00:00	
	12-11-2023	20:00:00	23:00:00	filtering events	3:00:00	
	13-11-2023	14:00:00	16:00:00	implementing filters for events	2:00:00	
	13-11-2023	19:00:00	20:00:00	fixing components file	1:00:00	
	14-11-2023	19:00:00	21:00:00	restrictions on filter input	2:00:00	
	15-11-2023	9:00:00	11:00:00	starting on search fucntion for events	2:00:00	
	15-11-2023	23:00:00	0:00:00	search function for title	1:00:00	
	16-11-2023	22:30:00	23:30:00	fixing merge	1:00:00	
	16-11-2023	21:00:00	22:00:00		1:00:00	
	17-11-2023	10:30:00	12:00:00	starting implementations of venues backend	1:30:00	
	19-11-2023	13:00:00	15:00:00	venues backend	2:00:00	
	24-11-2023	14:00:00	16:00:00	adding the venue controller and starting to add API	2:00:00	
	24-11-2023	22:00:00	0:00:00	Venues with API	2:00:00	
	25-11-2023	8:00:00	9:00:00	Venues with API	1:00:00	
	27-11-2023	12:00:00	14:30:00	Adding rating to venues + merge	2:30:00	
	6-12-2023	22:30:00	0:00:00	Hamburger menu	1:30:00	
	7-12-2023	0:00:00	2:00:00	Hamburger menu	2:00:00	
	7- 12-2023	14:00:00	16:00:00	hamburger icon interaction	2:00:00	
	9-12-2023	8:00:00	10:00:00	hamburger menu	2:00:00	
	9-12-2023	13:00:00	18:00:00	hamburger menu	5:00:00	
	10-12-2023	14:30:00	19:00:00	Mobile sidebar	4:30:00	
	10-12-2023	21:00:00	0:00:00	fixing hamburger menu functionalities	3:00:00	
	11-12-2023	0:00:00	1:30:00	fixing close notification function wit hamburger menu	1:30:00	
	12-12-2023	12:00:00	16:00:00	adding al functionalities to hamburger menu	4:00:00	
	12-12-2023	20:00:00	23:00:00	mobile homepage	3:00:00	
	13-12-2023	10:00:00	12:00:00	add event page	2:00:00	
	13-12-2023	13:00:00	15:30:00	add event page	2:30:00	
	13-12-2023	20:00:00	23:00:00	settings page	3:00:00	
	14-12-2023	14:00:00	15:00:00	concert page	1:00:00	
	14-12-2023	16:00:00	19:00:00	mobile functions	3:00:00	
	16-12-2023	22:00:00	23:59:00	mobile tickets	1:59:00	
	17-12-2023	15:00:00	20:00:00	mobile login, register, map and settings page	5:00:00	
	17-12-2023	21:30:00	23:00:00	mobile	1:30:00	
	18-12-2023	10:30:00	11:30:00	mobile	1:00:00	
	18-12-2023	22:30:00	23:59:00	mobile	1:29:00	
	19-12-2023	0:00:00	2:00:00	settings page	2:00:00	
	19-12-2023	15:30:00	19:00:00	account page	3:30:00	
	19-12-2023	20:00:00	23:59:00	hamburger, account and sidebar	3:59:00	
	20-12-2023	0:00:00	4:00:00	recording videos and preparing presentation	4:00:00	
	21-12-2023	15:00:00	16:00:00	making the report	1:00:00	
	22-12-2023	14:00:00	17:00:00	making the report	3:00:00	
	22-12-2023	20:30:00	23:00:00	making the report	2:30:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	
					0:00:00	

Figure 34: Dante his timesheet