



ÜSKÜDAR UNIVERSITY □ FACULTY OF ENGINEERING AND NATURAL SCIENCES

Azima

(Events Management System)

B.Sc. Thesis by:
Abdulaziz Kattan – Abdulaziz Faham

Department: Software Engineering (English)
Thesis Supervisor: Prof. Dr. Kristin Surpuhi Benli

June 2, 2024

Table of Contents

1. Introduction.....	1
2. Literature Review	1
2.1 Questionnaire	1
2.2 Researching Similar Products	3
3. Overall Description	6
3.1 Product Perspective.....	6
3.2 User Classes and Characteristics.....	6
3.3 Materials and Methods.....	7
3.3.1 User Interface and Diagrams Design	7
3.3.2 IDE: VS Code	8
3.3.3 Version Control.....	9
3.3.4 Frontend	10
3.3.5 Backend.....	17
3.3.6 Testing.....	20
4. Results and Discussion.....	22
4.1 Analysis of Project Outcomes and Achievements	22
4.2 Discussion of Challenges Faced and Lessons Learned.....	29
4.3 Implications for Future Work	30
5. References	31

1. Introduction

Azima is an Event Management System where a user can create, join, or browse different events. Upon opening the application, the user will be greeted by several groups that they can join, of which each offer different types of events that are filtered by the users' preferences. The user has the choice to browse or join any group they seek. After joining, the user is now a member of the group and will have full access to the event and the information provided by the admins.

2. Literature Review

In this chapter, we will be introducing the methods we used to gather the data that helped us develop Azima. The methods with the results will be explained and analyzed individually.

2.1 Questionnaire

We created a questionnaire that contains 13 questions using google forms [37]. The first couple of questions were added to help us best analyze our user and their environment, by asking about their age, location, preferences...

The next set of questions was added to best customize our app to fit our user expectations.

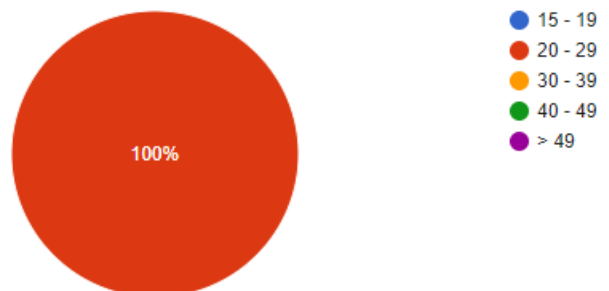
We have 9 closed questions and 4 open questions. We didn't want to have many questions, so the user won't be bored causing them to not complete the form. We tried keeping it brief and beneficial.

Link to our Google form survey: <https://forms.gle/WcW2i4UNRPm35tk59>.

In the upcoming figures you will come across questions where there is a given scale from 1 to 5. 1 will be the least and 5 will be the most, you will have extra explanation once you reach the questions.

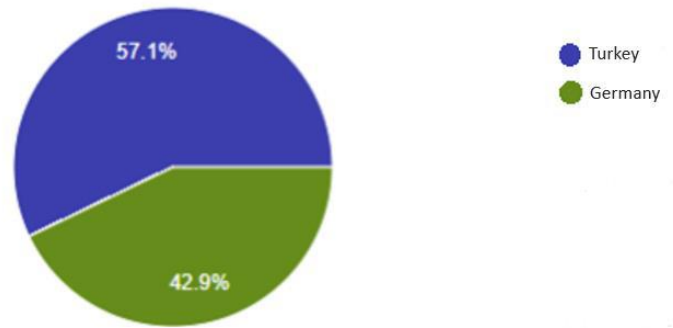
1. How old are you?

We added this question to know about our users' age range and customize our website depending on what is popular amongst similar-aged individuals. We concluded that 100% of our users are between the ages of 20 – 29, meaning that all our users are young adults.



2. Where do you live?

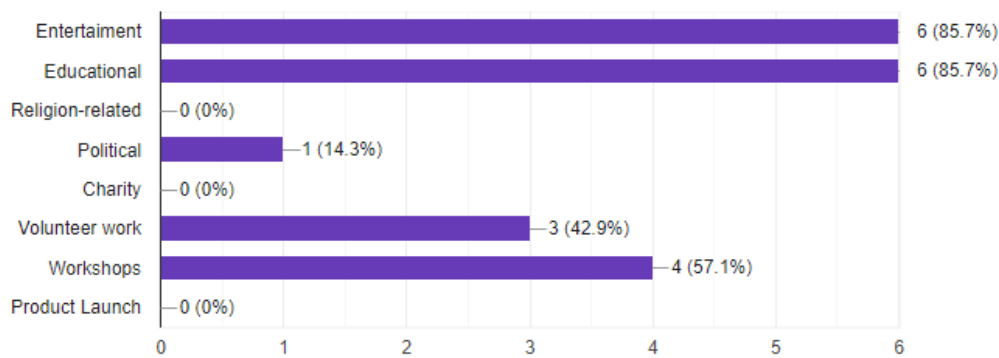
We came up with this question to know where our targeted users were located, making it easier for us to study their cultural habits and the usual events they would attend in their countries. The results were %57.1 of our users are in Turkey and the other %42.9 are in Germany. We studied German and the Turkish culture and the summary of it is the following:



1. Concerts halls and music venues: Turkish and German people enjoy attending concerts and live music performances. Concert halls like Zorlu PSM in Istanbul and Konzerthaus in Berlin. Live music is mostly performed in pubs and cafes.
2. Open-Air Festivals: Turkey and Germany host numerous festivals, especially in the summer months. Where a lot of people enjoy the sun doing different activities. For example, NEON Festival in turkey and Oktoberfest in Germany.
3. Exhibition Centers: Cities like Istanbul, Ankara, Berlin, Frankfurt, and Izmir have exhibition centers that host trade fairs, art exhibitions, and cultural events. The Istanbul Congress Center and TÜYAP Fair Convention and Congress Center are prominent venues for exhibitions and conferences.
4. Sports Arenas and Stadiums: Football (soccer) is a beloved sport in Turkey and Germany, people enthusiastically support their favorite teams. Stadiums like Allianz Arena in Munich, Vodafone Park in Istanbul, Signal Iduna Park in Dortmund, and Türk Telekom Stadium in Ankara are popular destinations for football matches and other sporting events.

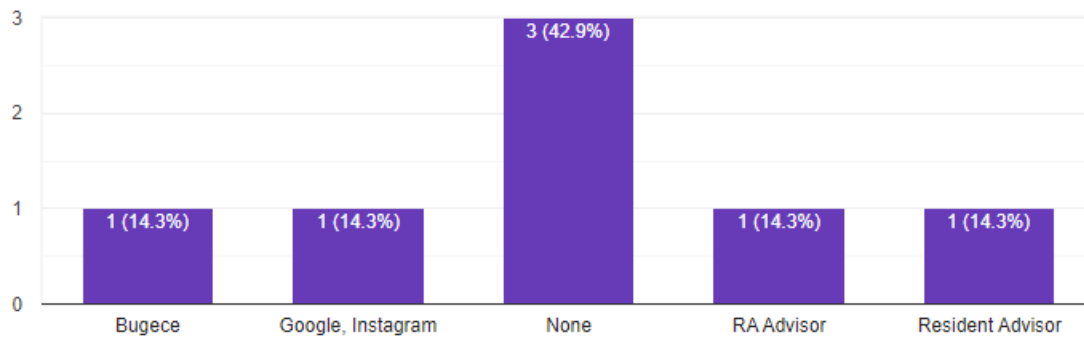
3. What categories of events are you most interested in?

This question helps us understand what our users' interests in events are which helps us to know what to focus on more. As we see here, we know that most of our users are into entertainment and educational events followed by workshops and volunteer work, which tells us to give the priority the same way.



4. What websites/mobile apps do you currently use to search for events?

This question was intended to get to know what type of systems users are used to. We then made a review study on them. We did a review study on both Bugece and Resident Advisor. Knowing that most of our users currently use no apps is good because it gives us flexibility while designing.

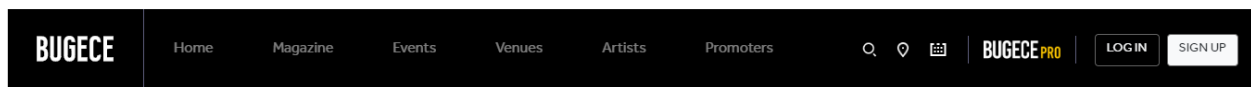


2.2 Researching Similar Products

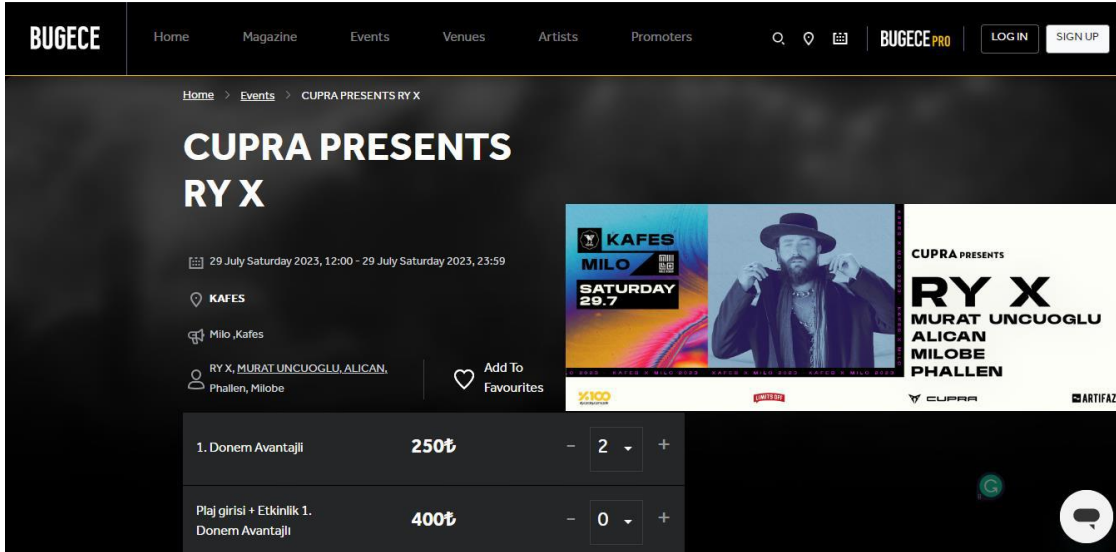
To conduct our research, we picked out the most used websites that people use, we also added a question in our questionnaire to know what people use most. We ended up with a couple of websites which we chose the top 2 from. Bugece and Resident Advisor.

2.2.1 Bugece

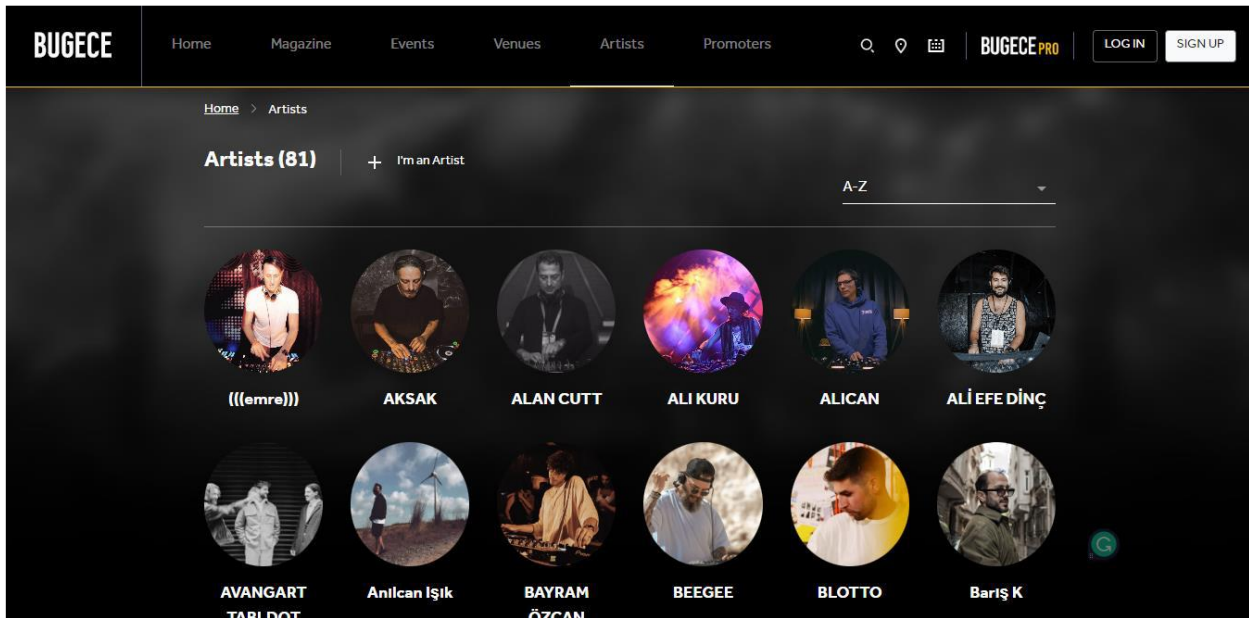
To start off, Bugece uses black and yellow colors as a theme, which we find convenient because it goes along with all the other event posters colors that will be added. It's nicer to the user's eyes.



Bugece has 2 buttons on the right for logged-out users so they can either log in or sign up. We applied the same method on our website. It also has its filters in the header which is a bit of an overload for the header. Therefore, we avoided using this feature.



When we click on an event, this page shows up. Multiple views are not convenient on this page, starting with the way the image is presented. We implemented instead a blurred background image. Moreover, adding icons next to the information was a useful feature we adopted in Azima.



This is the artist page on the website, the bubbles idea is good looking and creative. We implemented that on our homepage for the groups that'll be shown to the user.

From Bugece, we implemented:

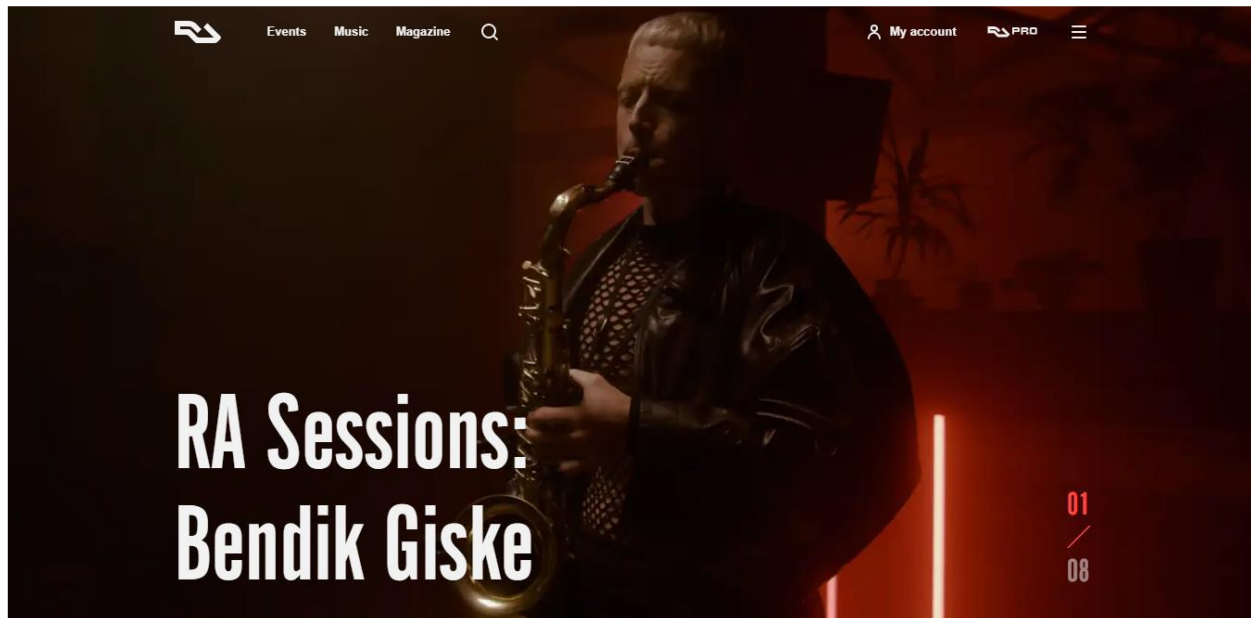
1. Header log in & Sign-Up buttons in the header.
2. Adding icons before certain types of info that describes it.
3. Circular view for groups.
4. Use a neutral palette.

We will avoid:

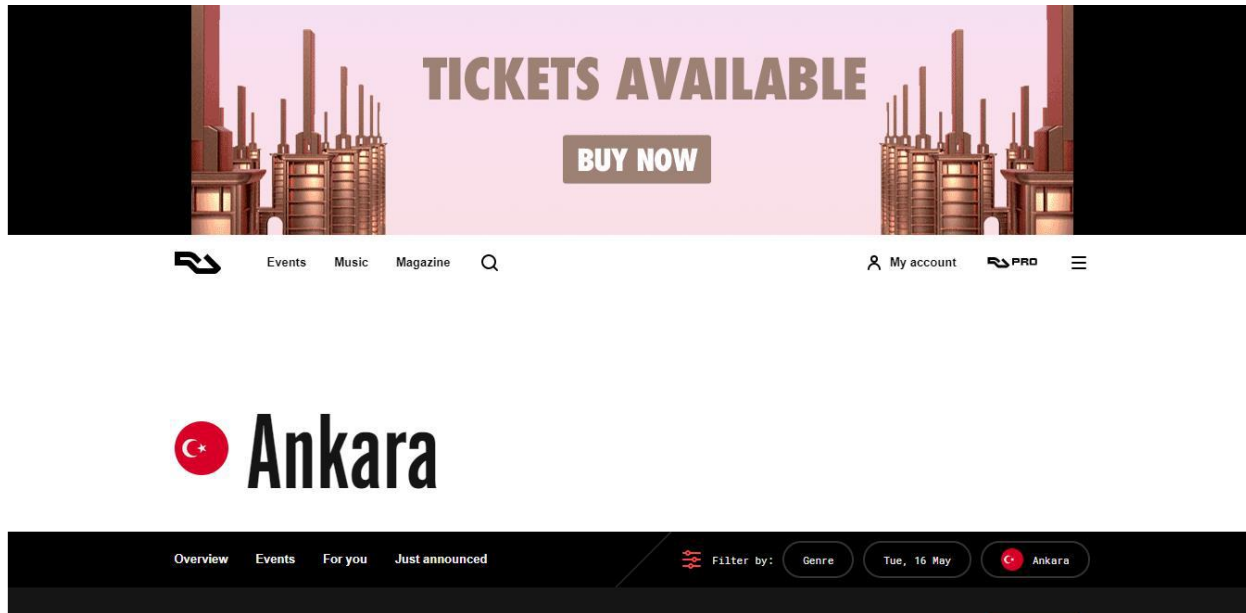
1. Filters and overloading options in the header.
2. Adding unnecessary info.

2.2.2 Resident Advisor (RA)

RA took personalization to be an essential part of their website design as they added a mini video as the background in their website. Their header is simple and merged in it, it is not overloaded with items which is more convenient.



In their events page, they have a big area to display an ad and another one to show the region of the shown events, which is unnecessary space for when a user opens a new page, the region is already shown in the header as well.



When you click on an event to see its details this is the page that appears, the blurred background is a precise addition. The way the data is presented is simple and clear. Providing the data the users care about knowing.

From RA we will implement:

1. A simple and merged header.
2. Design from the event page.

We will be avoiding:

1. Having unnecessarily spaces on our website.
2. A large footer.

3. Overall Description

3.1 Product Perspective

Our system is inspired by other event platforms, but we've made it easy for anyone to create or join interest-based groups and become part of existing events.

3.2 User Classes and Characteristics

Fresh User

An Individual user is not a part of any group or event yet. Hence, they can explore the public groups without having access to them until they decide to join one.

Group Member

A group member is a user that joined at least one group. They could explore the events inside the groups they joined, attend them if they wanted, and participate in making the events.

Group owner

A group owner is a user that created a group to organize their own events allowing people to join them. They can create events, edit group details, assign admins, ban members, request help from people, and delete the group.

Group Admin

A group admin is a group member that has been assigned by the group owner. They can do what the owner gave them access to.

3.3 Materials and Methods

The materials and methods we used to develop this web application are going to be explained below separately:

3.3.1 User Interface and Diagrams Design

During the UI design phase, we employed various methods to effectively sketch and visualize our ideas. Initially, we used hand-drawn sketches to outline the basic layout and to further enhance our design process, we utilized digital tools such as wireframing and prototyping software.

1. Lucidchart

Lucidchart is an online diagramming tool that simplifies visual content creation, including flowcharts, mind maps, and organizational charts, with an intuitive interface and a vast library of shapes and templates [9].

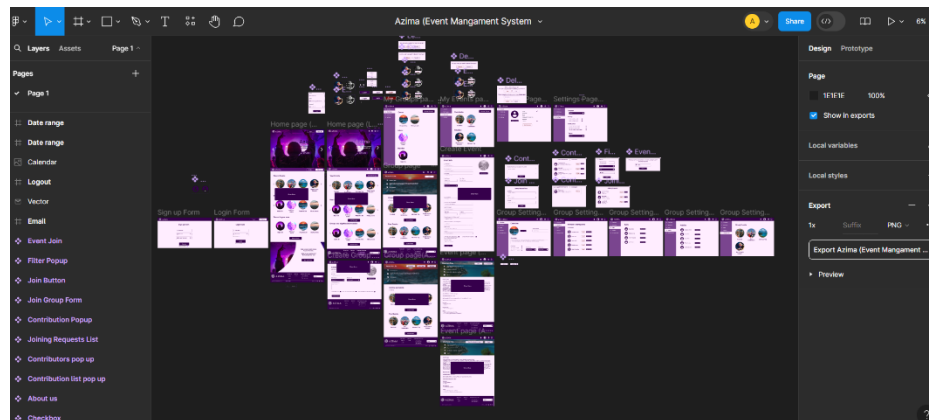
For our project, we used Lucidchart to create the data flow chart and entity-relationship diagram (ERD) [9]. Our database structure could be precisely represented and the relationships between various entities could be easily visualized thanks to Lucidchart's wide shape library and user-friendly interface [9].

[LucidCharts: Database design](#)

2. Figma

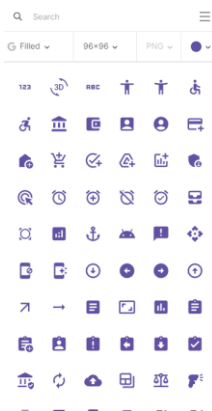
Figma is a powerful web-based design tool that allows teams to collaborate in real-time on creating user interfaces and prototypes [10]. A favorite among UI/UX designers, it provides a broad variety of design elements, a user-friendly interface, and strong collaborative capabilities.

Figma gave us an organized and clear vision to direct our development process, which we utilized as a guide when implementing our frontend design. Figma proved to be a useful tool. However, we had to make some changes during implementation to better suit the requirements of our project. It enabled us to establish a solid base and guarantee that our frontend design was easy and intuitive to use [10].



3. Material Design Icon

The Material Design Icons plugin for Figma provides a comprehensive library of icons that adhere to Google's Material Design guidelines [11]. With the help of this plugin, we could quickly find and use scalable, high-quality icons in our design.



3.3.2 IDE: VS Code

VS Code was chosen as our IDE due to its dynamic features, versatility, and cross-platform compatibility [12]. It offers a range of extensions, enhances productivity with its intuitive user interface, and supports syntax highlighting, code completion, and error detection.

We are including the extensions that was used in Azima and provided by VS Code below:

1. ES7+ React/Redux/React-Native snippets v4.4.3

Extensions for React, React-Native and Redux in JS/TS with ES7+ syntax. Customizable. Built-in integration with prettier [43].

2. JavaScript (ES6) code snippets v1.8.0

This extension contains code snippets for JavaScript in ES6 syntax for Vs Code editor (supports both JavaScript and TypeScript) [38].

3. Prettier - Code formatter v10.4.0

Prettier is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary [33].

4. Error Lens v3.18.0

Improve highlighting of errors, warnings, and other language diagnostics [35].

5. Tokyo Night v1.0.6

A clean Visual Studio Code theme that celebrates the lights of Downtown Tokyo at night [36].

6. vscode-pets v1.25.1

Puts a small, bored cat, an enthusiastic dog, a feisty snake, or a rubber duck in your code editor to boost productivity [34].

3.3.3 Version Control

Version control was crucial in Azima as it allowed us to track changes and collaborate effectively. By using a version control system, we could manage multiple contributors, resolve conflicts, and maintain a history of all modifications.

1. Git v2.40.1

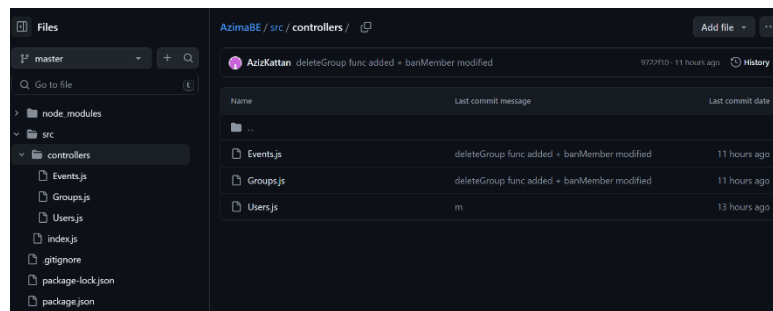
Git is a version control system that helps developers keep track of changes in their code over time. It allows multiple people to work on the same project simultaneously without overwriting our work [14].

Git include:

- Collaboration: Multiple team members can work on different parts of a project at the same time.
- History Tracking: Git records changes to the code, so you can see who made what changes and when.
- Reversibility: If something goes wrong, you can easily revert to a previous version of the code.

2. Github

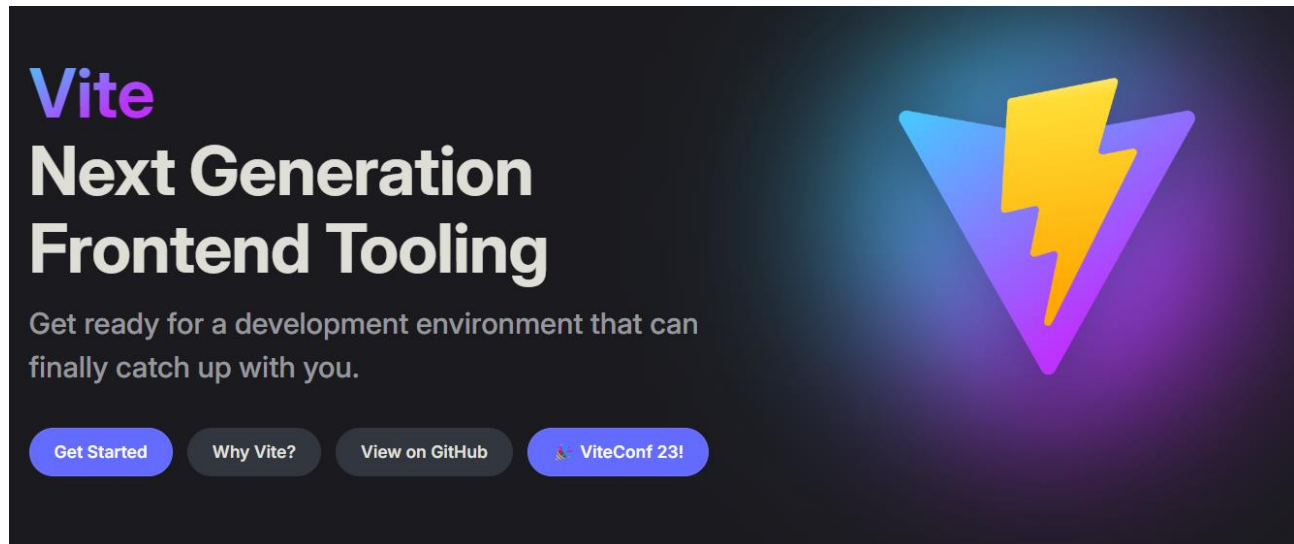
GitHub is an online platform that hosts Git repositories, making it easier for developers to collaborate on projects. It provides tools for version control, project management, and code review [13].



3.3.4 Frontend

After researching the available technologies that align with the nature of our web application and meet our design requirements, we concluded that the following technologies are suitable:

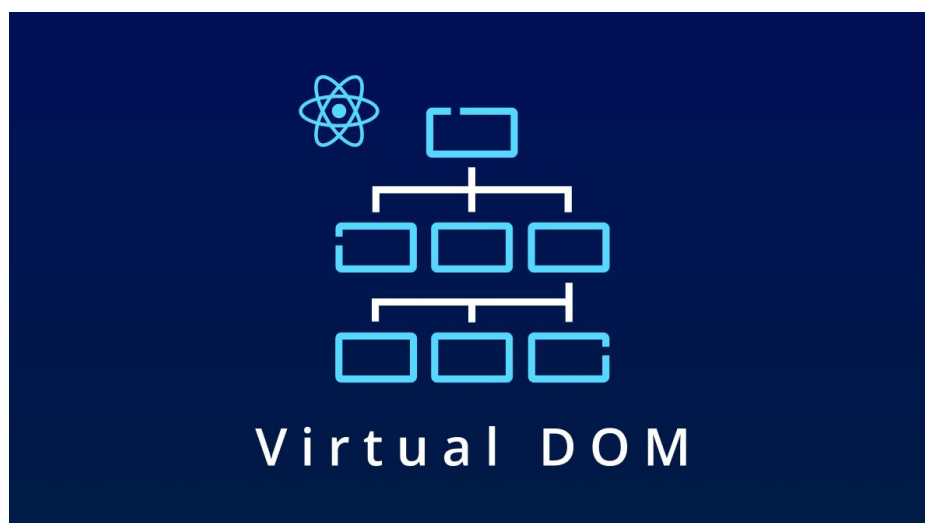
1. Vite v5.0.8



Vite is a modern tool for building web applications. It makes development faster with instant updates and minimal setup. For production, it optimizes and bundles your code efficiently. Vite supports popular frameworks like Vue, Svelte and React that we used to develop our application and will be covered in the below section [15].

2. React v18.2.0

Facebook's React is a popular JavaScript library for building user interfaces, particularly for single-page applications. It enables developers to create large web applications that update and render efficiently in response to data changes. React's virtual DOM [28] optimizes rendering by updating only the parts that have changed, making it fast and responsive. Its ecosystem includes tools like React Router [36] and Redux, making it a popular choice for front-end development [26].



Why did we use React:

- **Component-Based Architecture:** React's reusable components improve maintainability and speed up development, which is important for features like posts, notifications, and user profiles in social web applications.
- **Efficient Rendering with Virtual DOM:** React's virtual DOM ensures a quick and seamless user experience for real-time changes and dynamic content feeds by optimizing performance by updating only modified portions of the UI [28].

```
    <EventSlider object={events} label="Your Events" />
  )}
</>
)}
{groupsData.map(({ id, data, label }) => (
  <EventSlider key={id} object={data} label={label} />
))}
</div>

<Footer />
```

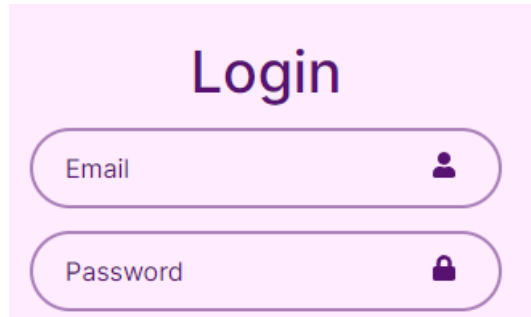
The following libraries were used during our development:

1) react-dom v18.2.0

React-DOM includes updates that make your React apps faster and more efficient. It supports concurrent rendering, which helps your app stay responsive by preparing multiple UI versions simultaneously. It also features automatic batching to minimize re-renders and improved server-side rendering (SSR) for quicker page loads [27].

2) react-icons v5.0.1

React-Icons is a library that provides a comprehensive collection of popular icons to use in React applications. It includes icons from various sets such as Font Awesome, Material Design, and others [28].

A login form with a light purple background. At the top, the word "Login" is centered in a large, bold, dark purple font. Below it are two rounded rectangular input fields. The first field is labeled "Email" in a dark purple font and has a small dark purple person icon on its right side. The second field is labeled "Password" in a dark purple font and has a small dark purple padlock icon on its right side.

3) react-router-dom v6.21.3

React-Router-Dom is a powerful library for managing navigation and routing in React applications. This version offers a simplified and more intuitive API for defining routes, handling dynamic routing, and navigating between different parts of the application [29].

```
<AuthProvider>
  <BrowserRouter>
    <Routes>
      <Route index element={<HomePage />} />
      <Route path="/Home" element={<HomePage />} />
      <Route path="/Signin" element={<Signin />} />
      <Route path="/Signup" element={<Signup />} />
      <Route path="/CreateGroup" element={<CreateGroup />} />
      <Route path="/GroupPage/:id" element={<GroupPage />} />
      <Route path="/CreateEvent/:id" element={<CreateEvent />} />
      <Route path="/EventPage/:id" element={<EventPage />} />
      <Route path="/Settings" element={<Settings />} />
      <Route path="/GroupSettings/:id" element={<GroupSettings />} />
      <Route path="/GroupMembers/:id" element={<GroupMembers />} />
      <Route path="/UserProfile/:id" element={<UserProfile />} />
    </Routes>
  </BrowserRouter>
</AuthProvider>
```

4) react-select v5.8.0

React-Select is a versatile library for creating custom dropdown menus and select components in React applications. It provides a wide range of features such as single or multi-select options, asynchronous loading of options, and customization of styles and behavior [30].

```
<Select
  className="dropdown"
  id="dropdown"
  value={value && value?.length > 0 ? value.map((val) => val) : []}
  name={label}
  options={list}
  isMulti={multiSelect}
  styles={colourStyles}
  onChange={onChange}
/>
```

3. TypeScript v5.2.2

TypeScript is a strongly typed superset of JavaScript that compiles to plain JavaScript. By including static type definitions, it improves the development process by enabling developers to identify errors during compilation as opposed to runtime. This produces code that is stronger and easier to maintain [16].

We used TypeScript in our project to detect errors at an early stage of development, which greatly decreased the possibility of problems at runtime. Through the usage of TypeScript's static type-checking, we were able to find and fix such issues before they had an impact on the user experience or performance of the application. This proactive approach to error management simplified the development process and enhanced the quality of the code [16].

```
interface User {
  ID: number;
  email: string;
  name: string;
  surname: string;
  password: string;
  birthdate: string;
  is_email_notifications_on: boolean;
  is_email_private: boolean;
  is_name_private: boolean;
  is_sms_notifications_on: boolean;
  is_theme_dark: boolean;
  is_updates_notifications_on: boolean;
  language: string | null;
  profile_image: string;
  username: string;
}
```


The figure above shows a user object type deceleration using typescript to handle type errors that happen in plain JavaScript.

4. CSS3

CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation of a document written in HTML or XML. It enables developers to separate content from design by defining how elements should be displayed on a web page, including layout, colors, fonts, and animations. CSS enhances user experience by making web pages visually appealing and responsive, allowing for flexible and adaptive designs that work across various devices and screen sizes. By using CSS, developers can maintain consistent styling and improve the maintainability of their web applications [17].

5. Axios

Axios is a popular JavaScript library used in front-end development for making HTTP requests. It simplifies the process of sending asynchronous requests to APIs, allowing developers to easily fetch data, post data, and handle responses. With Axios, managing responses, handling errors, and configuring request headers becomes straightforward [18].

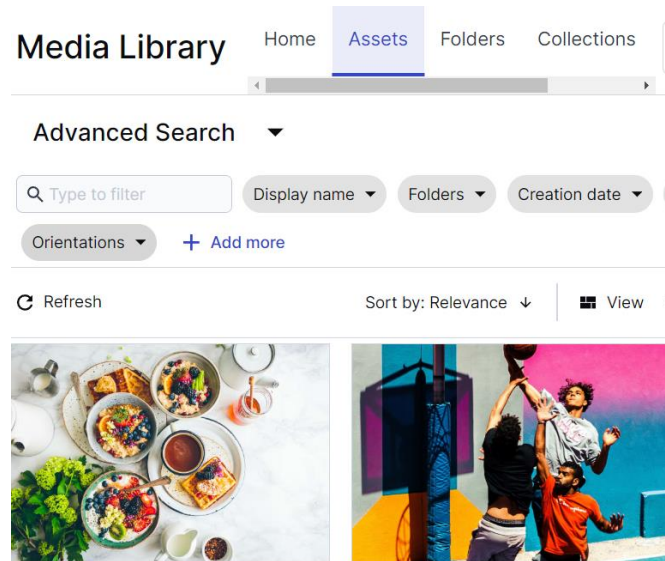
The figure (XXX) shows an Axios request to create and event, the server side then inserts the new event into the database and returns an event object that is stored in response.data.

```
const createEvent = async (eventData: any) => {  
  try {  
    const response = await axios.post(  
      "http://localhost:9000/createEvent",  
      eventData  
    );  
  
    setEvent(() => response.data);  
  
    navigate(`/EventPage/${response.data.event_id}`);  
  } catch (error) {  
    console.error(error);  
  }  
};
```

6. Cloudinary/url-gen v1.19.0

Cloudinary's Programmable Media is a software-as-a-service (SaaS) solution for developers. Upload, Manage and Analyze, Optimize and Deliver, Transform and Customize, Programmable Media offers APIs that let you automate the whole lifecycle of your image and video assets [22].

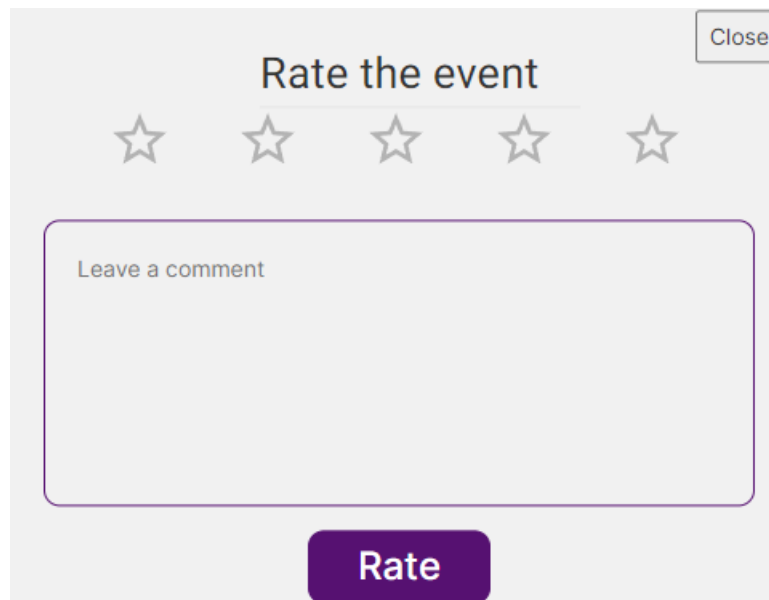
We applied Cloudinary in Azima to generate unique URLs, save them in the cloud it offers, then to be inserted into the database and used across the app.



7. MUI Material v5.15.18

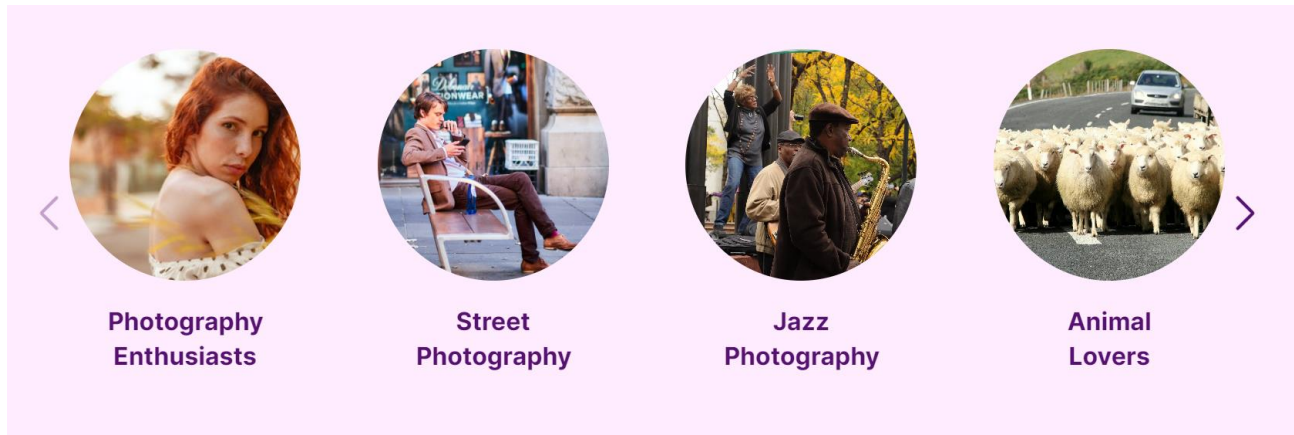
MUI offers a comprehensive suite of free UI tools to help you ship new features faster. Start with Material UI, our fully loaded component library, or bring your own design system to production-ready components [19].

In Azima we implemented the rating system using MUI rating components that allows us to store values in states then into PostgreSQL after selecting star count by user. The figure below displays the component style when rating an event.



8. Swiper v11.1.1

Swiper is a lightweight, modern JavaScript library for creating touch-friendly web sliders and carousels. It supports various use cases, features responsive design, navigation controls, and transition effects, making it an excellent choice for web development [31].

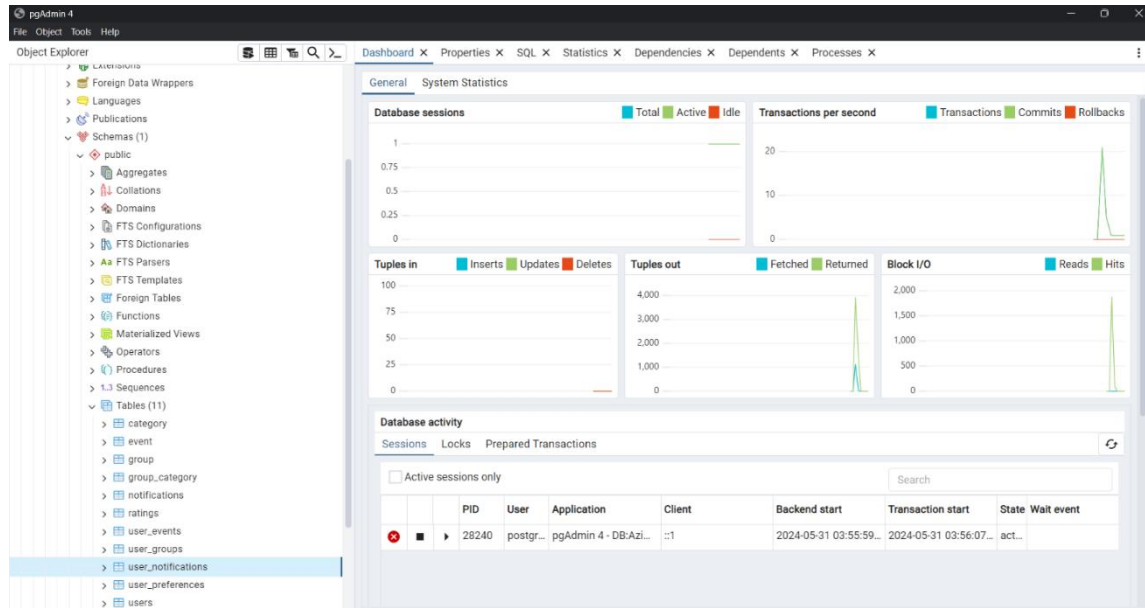


3.3.5 Backend

After researching the available technologies that align with the nature of our web application and meet our functional requirements, we concluded that the following technologies are suitable for the backend/server side.

1. PostgreSQL v16.2

PostgreSQL is a powerful, open-source database system used to develop our web application. Known for its scalability and reliability, PostgreSQL handles complex SQL queries and diverse data efficiently. Its support for advanced data types and strong community backing make it an ideal choice for the backend of Azima [1].



We used pgAdmin4 v8.2. Which has a user-friendly GUI that makes the database operations and handling much easier and manageable.

2. SQL

In our project, we utilized SQL to manage and manipulate data within our PostgreSQL database. SQL, the standard language for relational database management, enabled us to perform complex queries and efficiently handle data retrieval and updates. We used SQL to define our database schemas, create tables, and establish relationships between different entities in our system. This allowed us to implement essential features such as data filtering, sorting, and aggregation, which are crucial for generating insightful reports and analytics. Additionally, SQL played a key role in maintaining data integrity and consistency using constraints, indexes, and transactions. By utilizing SQL's robust capabilities, we ensured the efficiency and reliability of our database operations, forming a solid foundation for our application's data layer [2].

3. node.js v1.89.1 / express.js v2.8.5

We chose to combine Node.js with Express.js in our project because of their complimentary advantages for creating scalable and reliable online applications. We were able to effectively execute JavaScript code on the server-side thanks to the runtime environment provided by Node.js. Its non-blocking I/O approach ensured excellent performance by meeting our demand for managing concurrent processes [3] [4].

```
import express from "express";
```

```
21  const app = express();
22
23  app.use(express.json());
24  app.use(cors());
25
26
27  app.post("/login", authorizeLogin)
28  app.post("/register", register)
```

We expedited the development process by utilizing Express.js's robust features for HTTP request processing, middleware management, and routing. The flexible and basic design of Express.js enabled us to organize our application in a way that best suited our needs, and its vast ecosystem of middleware and plugins made it easier to incorporate crucial features like data validation and authentication. Node.js and Express.js worked together to give our project a strong basis, enabling quick development and ensuring the reliability and scalability of our web application [3] [4].

4. CORS v2.8.5

In our project, we made use of the CORS (Cross-Origin Resource Sharing) method in our Node.js/Express.js backend, especially in the development stage, to enable smooth communication with our Postman agent for testing needs. The Postman agent was able to send requests to our backend API endpoints without running into cross-origin limitations thanks to CORS. By integrating the cors middleware into our Express.js application, we configured the necessary headers to authorize cross-origin requests from the Postman agent. This streamlined our testing process, allowing us to validate API functionality effectively while ensuring the security of our web application [6].

```
import cors from "cors";
```

```
app.use(express.json());
app.use(cors());
```

5. pg-promise

We used the pg-promise library, integral to our project's backend development. This library, employed within our Node.js application, offers essential methods like none, one, oneOrNone, many, and manyOrNone, optimizing database interactions. These methods ensure precise querying, accommodating scenarios with varying result sets while maintaining data integrity and robust error handling. It also uses the tryAndCatch construct to handle asynchronous tasks with ease [5].

Ex:

```
3  import pgPromise from "pg-promise";
```

```

5  const db = pgPromise()('postgresql://postgres:[REDACTED]@localhost:5432/AzimaMainDB')
6
18  try {
19    const userPreferences = await db.anyOrNone('SELECT category_id FROM user_preferences WHERE user_id = $1', user.ID)
20    .then(data => data.map(row => row.category_id));
21
22    if (!userPreferences || userPreferences.length === 0) {
23      return res.status(404).json({ user, msg: "User preferences not found" });
24    }
25
26    const groups = await db.anyOrNone(`
27      SELECT DISTINCT g.*
28      FROM "group" g
29      JOIN group_category gc ON g.group_id = gc.group_id
30      WHERE gc.category_id IN ($1:csv)
31      `, [userPreferences]);
32
33    return res.status(200).json({ user, userPreferences, groups: groups });
34  } catch (error) {
35    console.error("Error fetching user preferences:", error);
36    return res.status(500).json({ msg: "Internal server error" });
37  }

```

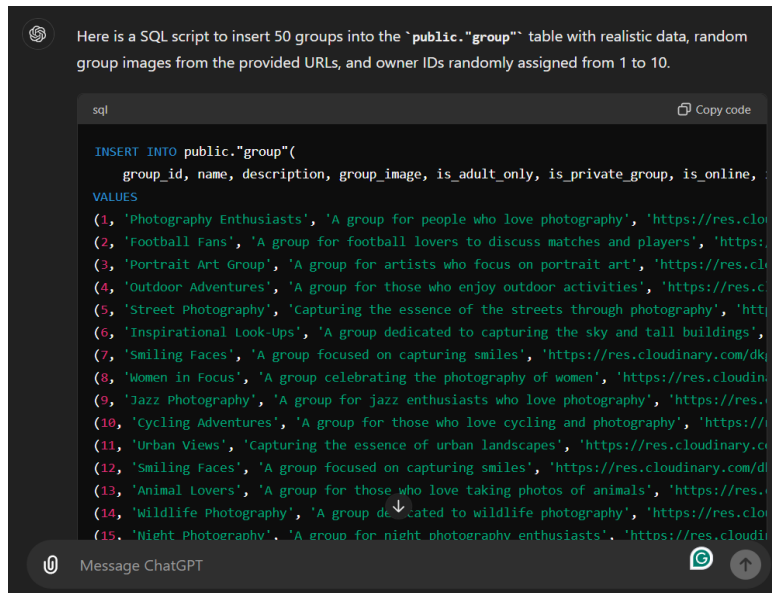
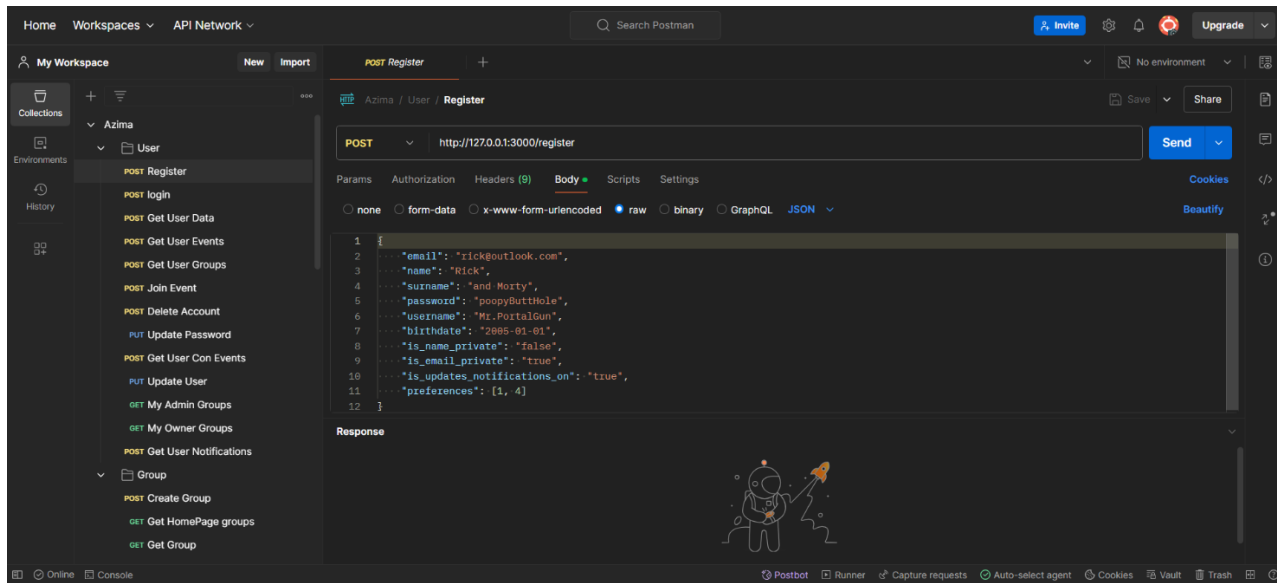
red: tryAndCatch() method

yellow: queries using manyOrNone()

Overall, the strategic application of pg-promise enhances the efficiency, reliability, and maintainability of our backend codebase, highlighting its critical role in facilitating PostgreSQL database interactions within our project.

3.3.6 Testing

The testing phase involved a dual approach using Postman agent and ChatGPT to ensure comprehensive coverage of the application's functionality. Postman agent was used for manual testing of API endpoints, validating request-response cycles and evaluating system behavior in real-time. ChatGPT was used to automate the generation of test scenarios, covering various edge cases and scenarios [7] [8].



This hybrid testing methodology assessed the resilience and functionality of the backend infrastructure, ensuring high reliability and performance in the web application.

4. Results and Discussion

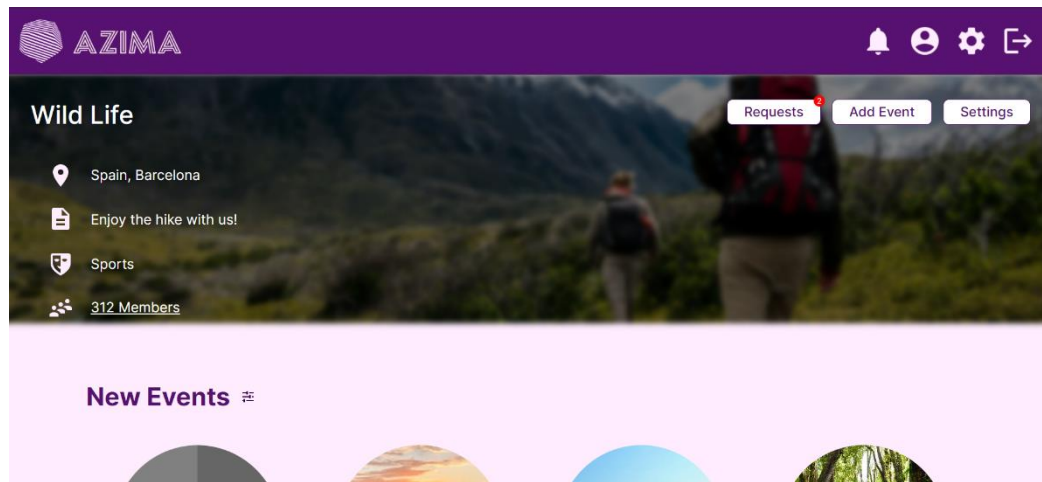
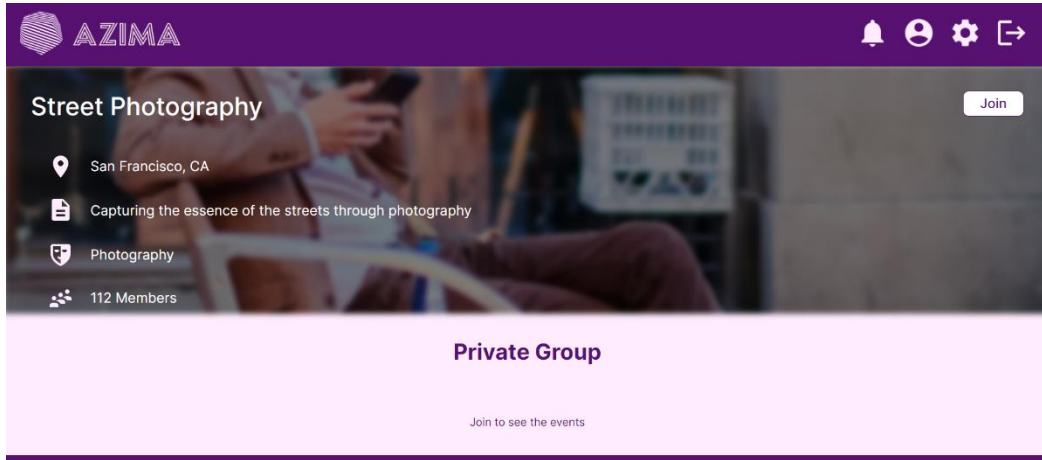
4.1 Analysis of Project Outcomes and Achievements

The development of the "Azima" event management system successfully met the project's primary objectives. Key achievements include:

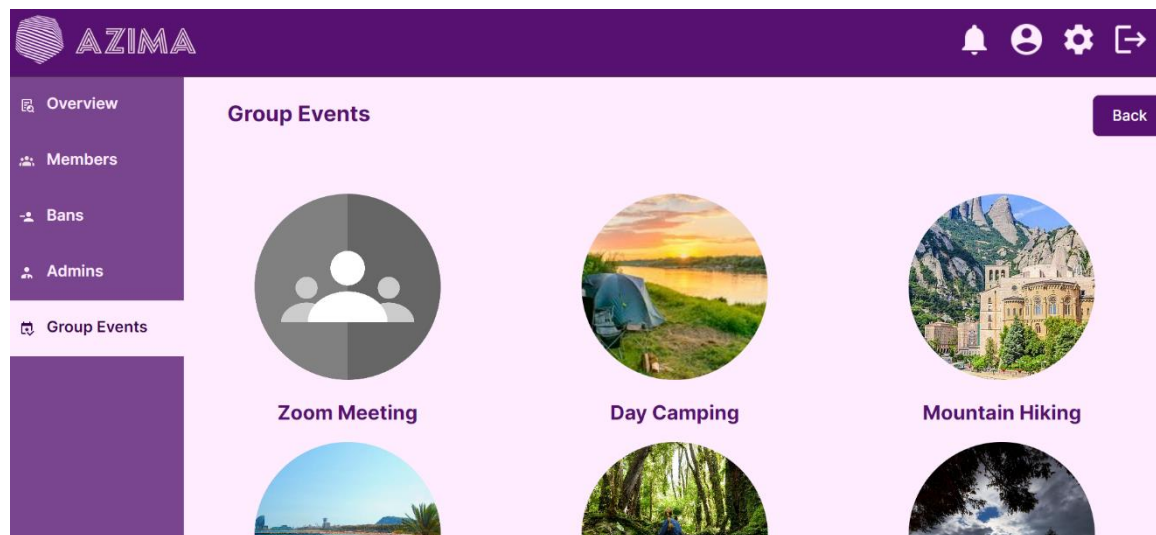
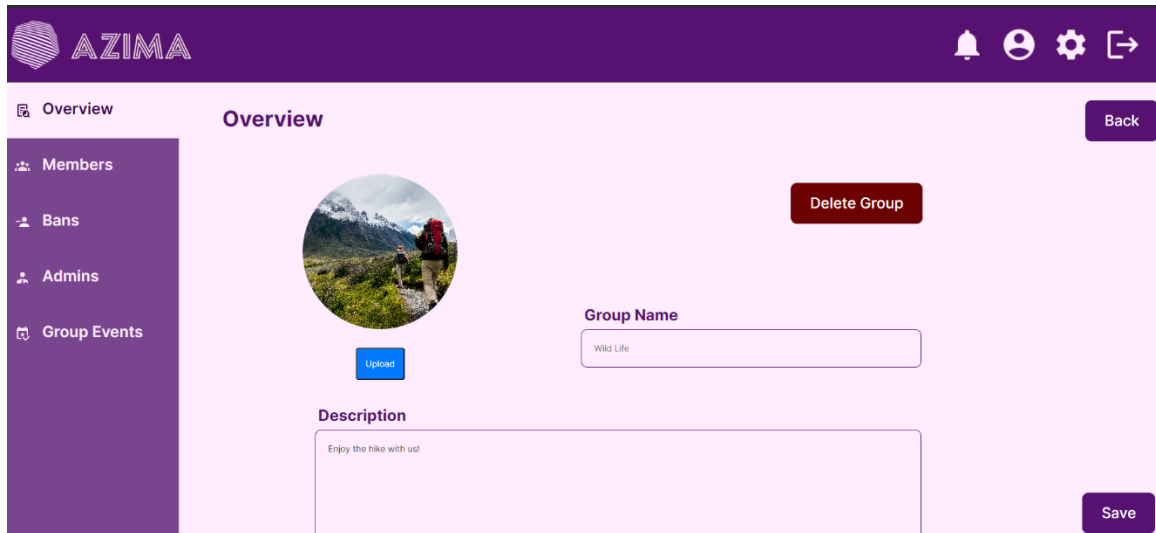
- **User Account Management:** Users can create and manage their accounts, including updating personal information and privacy settings (e.g., making name, surname, and email private).

The screenshot displays the 'Account' management page within the Azima system. The interface features a purple header with the 'AZIMA' logo and navigation icons. A left sidebar contains links for 'Account', 'My Groups', and 'My Events'. The main content area is divided into two sections. The top section, titled 'Account', contains input fields for 'Name' (filled with 'Aziz'), 'Surname' (filled with 'Faham'), 'Username' (filled with 'Elzo5'), 'Birthdate' (with a date picker set to 'mm/dd/yyyy'), and 'Email' (filled with 'test@test.com'). A profile picture of a man is shown with an 'Upload' button. The bottom section contains 'Email' (test@test.com), 'Password' (masked with asterisks), a 'Change Password' button, 'Interests' (checkboxes for Sports, Comedy, Education, and Religion, with Sports and Comedy selected), 'Privacy' settings (checkboxes for 'Hide my name' and 'Hide my email', both unchecked, and a checked checkbox for 'Send me notifications about updates'), and buttons for 'Delete Account' and 'Save'.

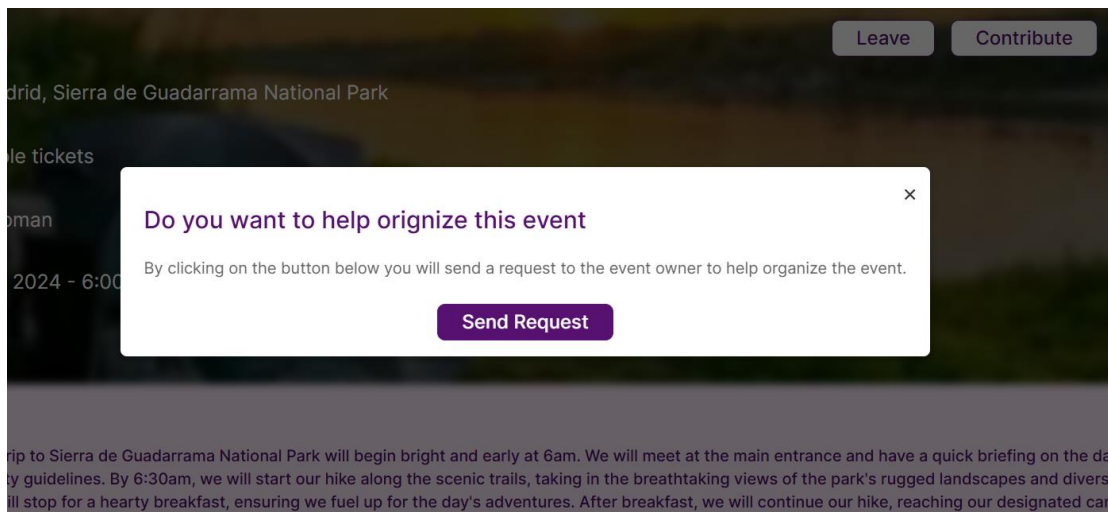
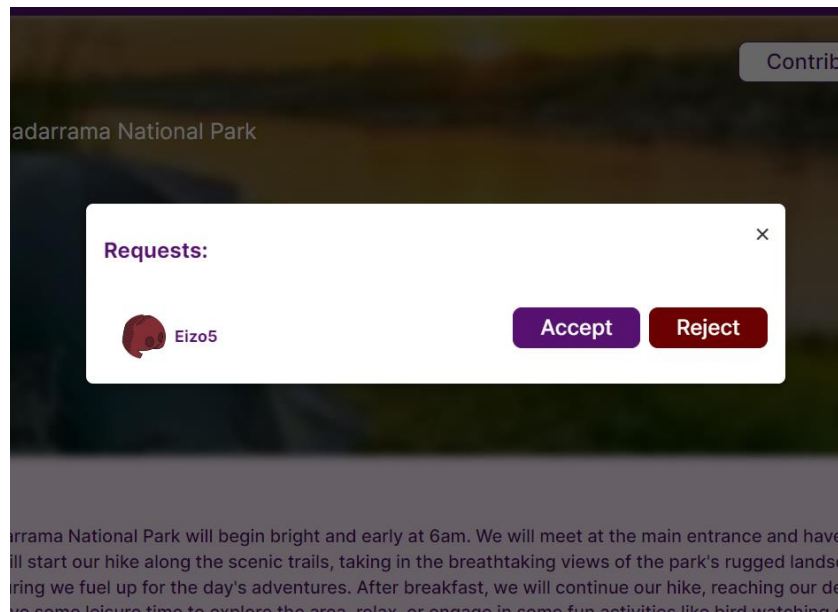
- **Group Management:** Users can view, join, and create groups. Groups can be public or private, offering flexibility in user engagement and content visibility.



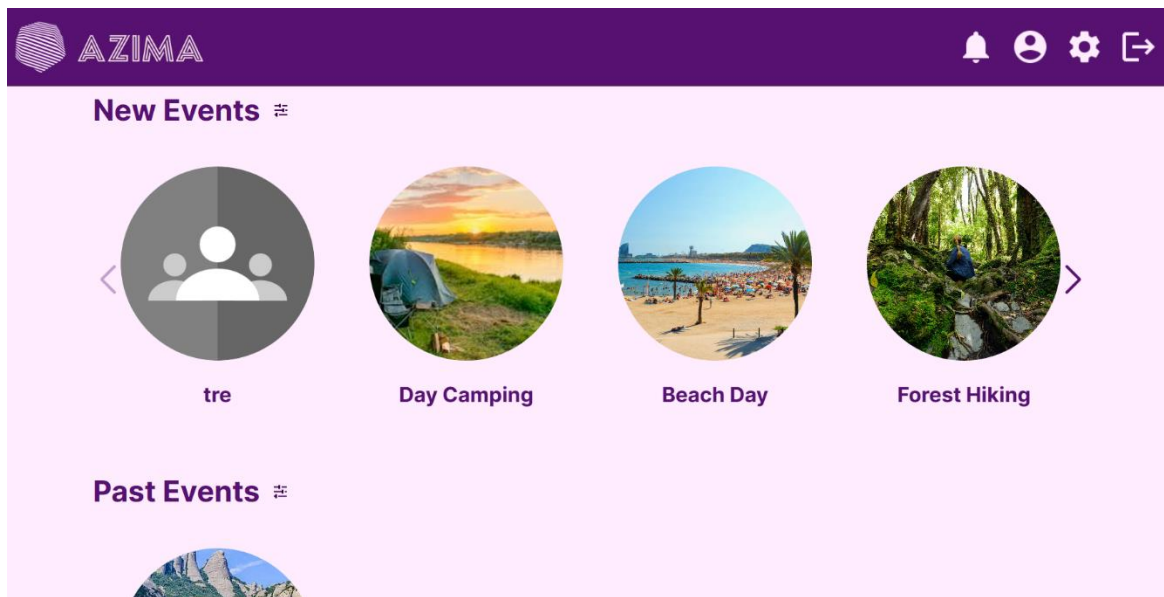
- **Role-Based Access Control:** The system supports multiple user roles within groups—group owner, admin, and member. Owners can assign admin roles, and members can request or be assigned the admin role. Admins and owners can create and manage events within their groups.



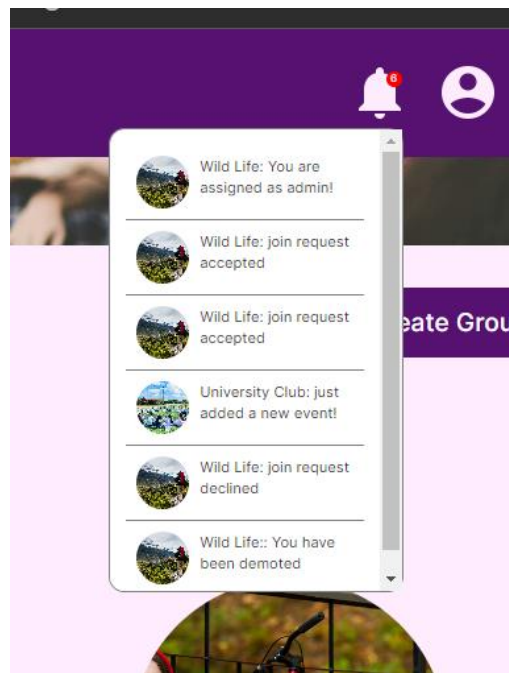
- **Event Management:** Users can create events within groups they own or admin. There is also a feature for assigning contributors to events, either by admin assignment or user request.



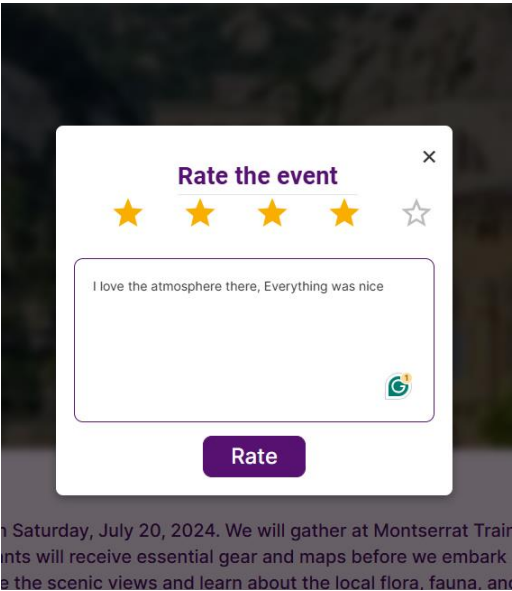
- **Public Group Display:** Non-registered users can browse random public groups on the homepage, encouraging engagement and potential registration.



- **Notification System:** An integrated notification system keeps users informed about new events, event deletions, role assignments (admin/contributor), and other relevant updates.



- **Event Rating System:** Users can rate events, providing feedback that can help improve future events.




Available tickets: 48

Return policy: No return

Guests: Naval Perez

Event Ratings




★★★★☆

Delete

Eizo5

0 seconds ago

I love the atmosphere there, Everything was nice




★★★★☆

charliedavis5

4 minutes ago

Great event, well organized and lots of fun!




★★★★☆

danawilson6

4 minutes ago

Had a fantastic time, met some wonderful people.



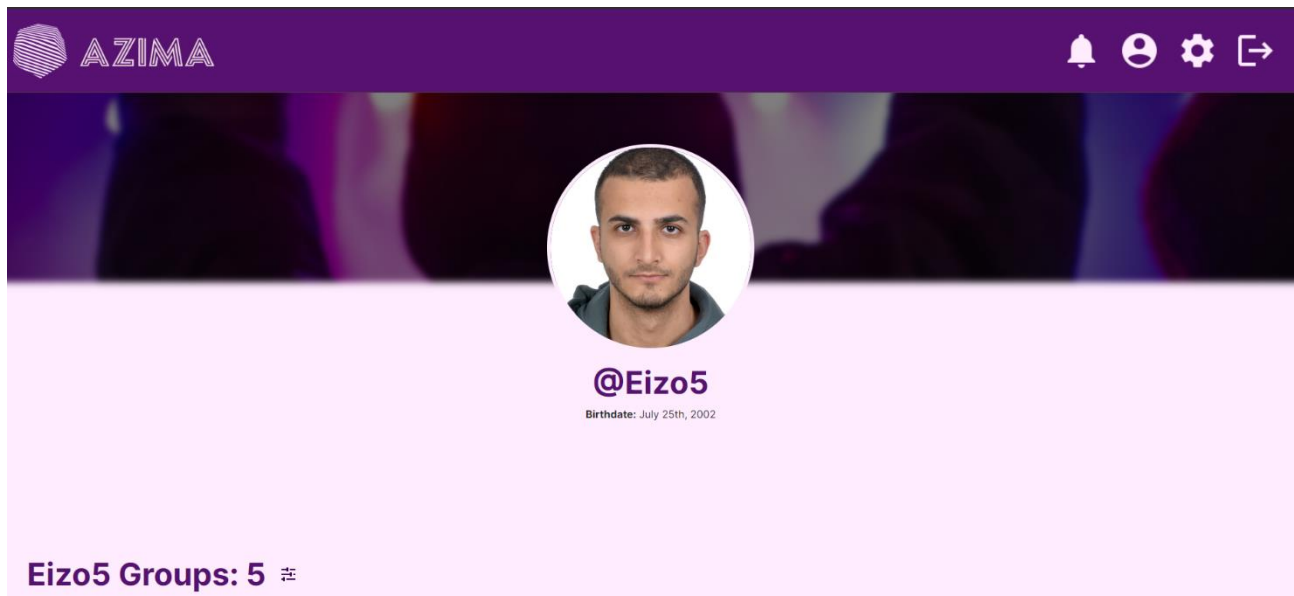
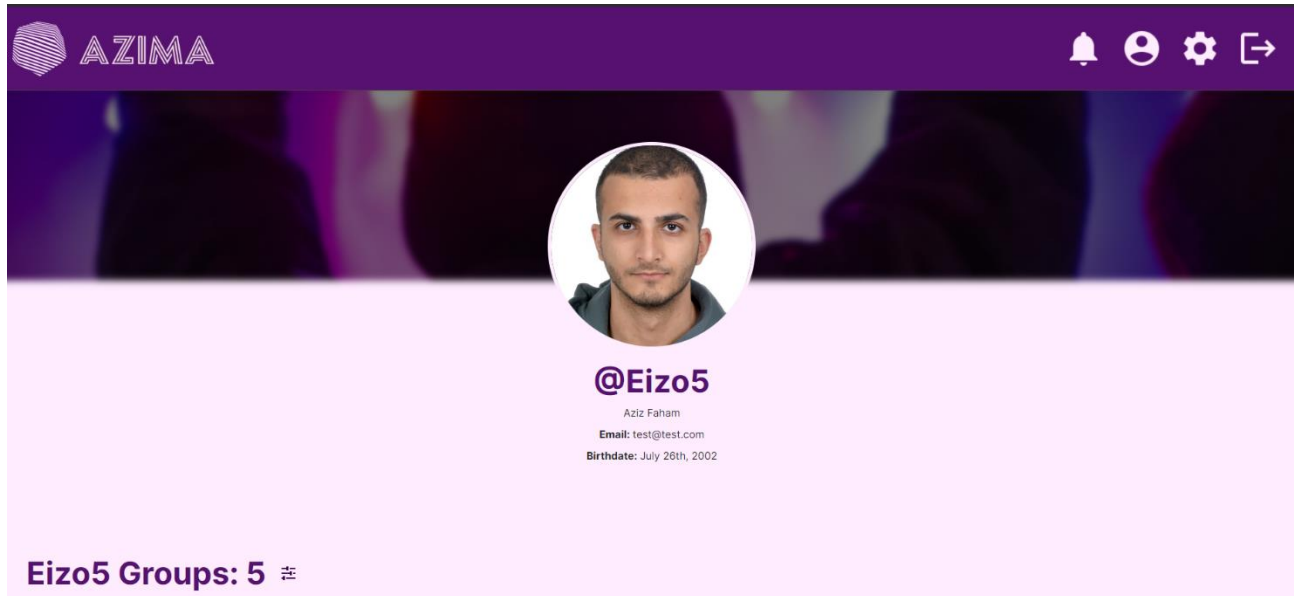
★★★★★

evemiller7

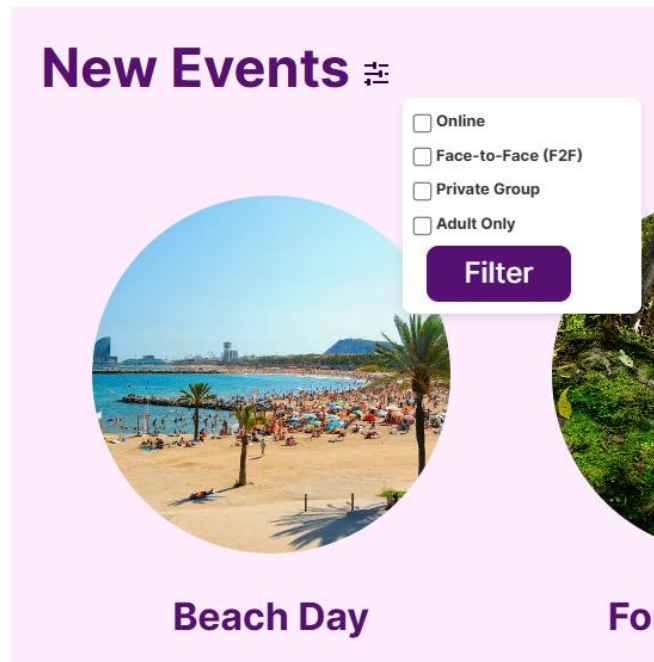
4 minutes ago

Amazing experience, looking forward to the next one!

- **Privacy Controls:** Users have control over the visibility of their personal information, enhancing privacy and security.



- **Group Filtering:** The system includes a group filtering feature on the homepage, allowing users to search and filter groups based on specific criteria such as group type (public or private), meeting format (face-to-face or online), and age restriction (+18 only). This enhances user experience by enabling users to quickly find groups that match their preferences and requirements.



4.2 Discussion of Challenges Faced and Lessons Learned

Throughout the development process, several challenges were encountered, and valuable lessons were learned:

- **Technical Integration:** Integrating various technologies (Node.js, Express.js, PostgreSQL, React, etc.) posed initial challenges, particularly in ensuring seamless communication between the frontend and backend. These were addressed through thorough testing and debugging.
- **Database Optimization:** Optimizing database queries to handle large datasets efficiently was crucial. Methods such as `one`, `findOne`, `many`, `any`, and `manyOrNone` in `pg-promise` were instrumental in managing different query scenarios. Proper indexing and query optimization techniques were also employed to enhance performance.
- **Role Management:** Managing user roles and permissions required careful planning and implementation to ensure users had the correct access rights. This involved creating a flexible system for role assignments and transitions (e.g., member to admin).

- **Notification System:** Implementing a comprehensive notification system that reliably informs users of relevant updates and changes was a significant challenge. Ensuring notifications were timely and accurate required extensive testing.
- **User Feedback:** Gathering and incorporating user feedback during the development process was invaluable. It helped identify usability issues and areas for improvement, ensuring that the final product met user expectations.
- **Filtering Implementation:** Implementing the group filtering feature on the homepage presented challenges due to the need to handle various filtering criteria (public or private, face-to-face or online, +18 only). To address this, we developed a dynamic query system capable of adapting to different input scenarios. This approach ensured that the filtering functionality was both efficient and flexible, providing accurate search results based on user-defined criteria.

4.3 Implications for Future Work

While the "Azima" system successfully achieved its initial objectives, there are several areas for future enhancement:

- **Advanced Analytics:** Incorporating advanced analytics to provide event organizers with deeper insights into attendee behavior, event performance, and marketing effectiveness.
- **Mobile Application:** Developing a dedicated mobile application to complement the web-based platform, providing users with a more tailored experience.
- **Integration with Social Media:** Enhancing the system's capabilities by integrating with social media platforms for easier event promotion and attendee engagement.
- **Scalability Enhancements:** Further optimizing the system to handle larger volumes of events and users, ensuring performance remains consistent as the user base grows.
- **On-Change Search Feature:** In the next version, we plan to implement an on-change search feature. This will allow users to see real-time search results as they type or adjust filters, improving the speed and efficiency of finding relevant groups. This enhancement aims to provide a more interactive and responsive user experience.

In conclusion, the development of the "Azima" event management system demonstrated the successful application of modern web technologies to create a functional, user-friendly platform. The project not only met its objectives but also provided valuable insights and opportunities for future development.

5. References

1. PostgreSQL Global Development Group. (2023). *PostgreSQL v16.2*. Retrieved from <https://www.postgresql.org>
2. ISO/IEC. (2016). *ISO/IEC 9075:2016 - Information technology - Database languages - SQL*. Retrieved from <https://www.iso.org/standard/76583.html>
3. Node.js Foundation. (2023). *Node.js v18.18.2*. Retrieved from <https://nodejs.org>
4. Express. (2023). *Express v4.19.2*. Retrieved from <https://expressjs.com>
5. Vitaly Tomilov. (2023). *pg-promise v11.6.0*. Retrieved from <https://github.com/vitaly-t/pg-promise>
6. Express.js. (2023). *CORS v4.19.2*. Retrieved from <https://www.npmjs.com/package/cors>
7. Postman. (2023). *Postman Agent*. Retrieved from <https://www.postman.com>
8. OpenAI. (2023). *ChatGPT v3.5*. Retrieved from <https://www.openai.com>
9. Lucid Software Inc. (2023). *Lucidcharts*. Retrieved from <https://www.lucidchart.com>
10. Figma, Inc. (2023). *Figma*. Retrieved from <https://www.figma.com>
11. Google. (2023). *Material Design Icons*. Retrieved from <https://fonts.google.com/icons>
12. Microsoft. (2023). *Visual Studio Code v1.89.1*. Retrieved from <https://code.visualstudio.com>
13. GitHub, Inc. (2023). *GitHub*. Retrieved from <https://github.com>
14. Software Freedom Conservancy. (2023). *Git v2.40.1*. Retrieved from <https://git-scm.com>
15. Vite. (2023). *Vite v5.0.8*. Retrieved from <https://vitejs.dev>
16. Microsoft. (2023). *TypeScript v5.2.2*. Retrieved from <https://www.typescriptlang.org>
17. W3C. (2018). *CSS3*. Retrieved from <https://www.w3.org/Style/CSS/>
18. Axios. (2023). *Axios v1.6.8*. Retrieved from <https://axios-http.com>
19. MUI. (2023). *mui/material v3.15.18*. Retrieved from <https://mui.com>
20. MUI. (2023). *mui/icons-material v3.15.18*. Retrieved from <https://mui.com>
21. Cloudinary. (2023). *cloudinary/react v1.13.0*. Retrieved from <https://cloudinary.com>
22. Cloudinary. (2023). *cloudinary/url-gen v1.19.0*. Retrieved from <https://cloudinary.com>
23. Emotion. (2023). *emotion/react v11.11.4*. Retrieved from <https://emotion.sh>
24. Emotion. (2023). *emotion/styled v11.11.5*. Retrieved from <https://emotion.sh>
25. Fontsource. (2023). *fontsource/roboto v5.0.13*. Retrieved from <https://fontsource.org>
26. Facebook Inc. (2023). *React v18.2.0*. Retrieved from <https://reactjs.org>
27. Facebook Inc. (2023). *react-dom v18.2.0*. Retrieved from <https://reactjs.org>
28. React Icons. (2023). *react-icons v5.0.1*. Retrieved from <https://react-icons.github.io/react-icons>
29. React Training. (2023). *react-router-dom v6.21.3*. Retrieved from <https://reactrouter.com>
30. Jed Watson. (2023). *react-select v5.8.0*. Retrieved from <https://react-select.com>
31. Swiper. (2023). *swiper v11.1.1*. Retrieved from <https://swiperjs.com>
32. Vite. (2023). *@vitejs/plugin-react v4.2.1*. Retrieved from <https://vitejs.dev>
33. ESLint. (2023). *eslint v8.55.0*. Retrieved from <https://eslint.org>
34. ESLint Plugin React Hooks. (2023). *eslint-plugin-react-hooks v4.6.0*. Retrieved from <https://reactjs.org>

35. ESLint Plugin React Refresh. (2023). *eslint-plugin-react-refresh* v0.4.5. Retrieved from <https://github.com/pmmmwh/react-refresh-webpack-plugin>
36. dsznajder. (2023). *ES7 + React/Redux/React-native snippets* (extension). Retrieved from <https://marketplace.visualstudio.com/items?itemName=dsznajder.es7-react-js-snippets>
37. Prettier. (2023). *Prettier - code formatter* (extension). Retrieved from <https://prettier.io>
38. Microsoft. (2023). *VsCode - Pets* (extension). Retrieved from <https://marketplace.visualstudio.com/items?itemName=tonybaloney.vscode-pets>
39. Alexander. (2023). *Error Lens* (extension). Retrieved from <https://marketplace.visualstudio.com/items?itemName=usernamehw.errorlens>
40. enkia. (2023). *Tokyo night* (extension). Retrieved from <https://marketplace.visualstudio.com/items?itemName=enkia.tokyo-night>
41. Google. (2023). Google Forms. Retrieved from <https://www.google.com/forms>
42. JavaScript (ES6) code snippets v1.8.0. (2023). Retrieved from [Visual Studio Code Marketplace](#).