Nazarbayev University Events Tracker

Final report

Group 52 Senior Project

Tamirlan Mendygaliyev

Rakhymzhan Turlybek

Yerdaulet Seidizhappar

Farkhad Amanbay

Alan Mukeyev

Alinur Mozhikov

Executive Summary

With so many different and exciting activities taking place on campus, the "Nazarbayev University Events Tracker" project offers a complete solution to improve campus participation and organize events more efficiently. Taking into account the difficulties presented by the wide variety of events—from social get-togethers to academic lectures—the project seeks to transform event monitoring in order to promote a more engaged and active campus community. Through the resolution of current issues related to event administration and monitoring, the "Nazarbayev University Events Tracker" system is a major step toward creating a more vibrant and integrated campus community. With careful planning, development, and execution, our team has created a solid solution that will improve the academic experience for all parties involved—faculty, staff, and students

.

Introduction

The exciting and interesting activities that a successful educational institution hosts are its lifeblood. These activities not only improve students' college experiences but also strengthen the campus community and sense of belonging. However, guaranteeing the smooth planning, advertising, and monitoring of these events is a difficult task with many different facets. There are more than 100 events happening each month at our campus. Ranging from dance parties to Ph.D. thesis presentations. It is hard to keep track of them. Students must follow different social media pages (Facebook, Instagram), keep track of emails, and join many online chat groups to keep in touch with recent events and social activities.

This project proposal aims to inspire and lay out a thorough plan to improve event monitoring at our university, which will ultimately result in a livelier and more sociable campus atmosphere. This project's main goal is to redesign our event tracking system in order to address the issues that currently exist and build a more active and involved campus community. Our project presents a solution to this problem by aggregating and recommending relevant events to users based on their interests. All these functionalities will be presented in a convenient map to ease the navigation and user experience. To make communication easier real-time chat per each event is also available for users.

Background information

The University Events Tracker app was inspired by a dynamic, all-inclusive solution created specifically for the hectic life on campus. Taking inspiration from popular event management platforms like Meetup and Eventbrite, this program aims to improve and accelerate how faculty, staff, and students engage with a variety of campus activities.

Meetup is a platform that facilitates in-person gatherings for a variety of interests and activities. It may be used by users to establish their own gatherings or locate and join local groups. Users may search events based on social causes, professional networks, hobbies, or interests. Meetup aims to unite individuals who have similar interests, promoting in-person interactions and community engagement.

Key Features:

- Discover and join groups based on interests.
- Organize or attend local events, from hobby groups to professional networking.
- Schedule and coordinate meetups with group members.
- Engage in discussions and share experiences within specific interest-based communities.

An online tool for managing events and selling tickets is called **Eventbrite**. It enables those who organize events to plan, promote, and manage events of all sizes. Users may choose from a wide range of events in several areas, such as business, sports, music, and more, from small gatherings to large conferences.

Key Features:

- Event creation and promotion tools for organizers.
- Ticketing and registration management.
- Discover and attend events across diverse categories.
- Attendee communication and event logistics support.

Both Meetup and Eventbrite offer unique approaches to event engagement. Meetup focuses on fostering in-person community connections and group meetups, while Eventbrite caters to event

organizers by providing a comprehensive platform for event creation, ticketing, and attendee management across different events.

There are also examples from local markets such as **Ticketon.kz and Zakazbiletov.kz**. They also provide information about events but they work for all countries and are focused on ticket selling.

Another Kazakhstan's example is a **Sxodim.kz** website which works as a news channel providing information about future events.

Literature Review

- Studies by Getz (2019) and Allen et al. (2020) emphasize the complexities of event planning, highlighting the need for systematic approaches to organizing successful events. Factors such as budgeting, logistics, marketing strategies, and stakeholder management play pivotal roles in ensuring event success.
- Research by Lee and Arcodia (2018) focuses on understanding attendee experiences and satisfaction levels at events. Attendee engagement, interaction, and perceived value contribute significantly to the success and impact of an event.
- The impact of technology on event management has been extensively researched (Gursoy et al., 2021). Studies highlight the role of digital platforms, mobile applications, and virtual event technologies in transforming event experiences.

Project Approach

- **System features**

1. **User Authorization and Authentication**
    - Account creation and login functionality for attendees.
    - The application verifies users and gives access
    - Different users have different permission level.
    - Users are stored in the Database(user password is encrypted).
2. **Event Creation**
    - Users are able to create events based on preferences.
    - After successful creation, the event is shown on a map.
    - Then, based on different scenarios the user is able to modify and change events. Such as setting rules, enabling chat, and so on.
3. **Event Details and Discovery**
    - Search and browse to discover the desired event
    - Filters for location, date, category
    - Attend the event(Subscribe to the event and get notifications)
4. **Interactive Maps**
    - Events are shown through the Yandex Map.
    - Navigation functionality and search are integrated.
5. **Push Notifications**
    - Event news, updates, and information are displayed and notified to the user.
6. **Feedback and Rating**
    - Users can give feedback on the event and rate it.
    - Other users can see all feedback.
    - High-rated events are shown first.
7. **Registration and Ticketing**
    - The event organizer has the functionality to manage the registration process by creating a QR code.
    - These online tickets are used to authorize users to attend the event and can be used for other needs.

8. **Analytics and Reporting**
    - The event application generates reports and analyzes all events.
    - The administrator has access to see all data.

9. **Recommendation System**
    - The application recommends similar attending events for users.
    - Using collaborative filtering we were able to suggest event to users with similar efforts

- **Requirements and Functionalities**

The incorporation of several services encourages us to utilize microservice architecture to achieve high performance, availability and scalability. This is mostly possible because of the independent nature of each service within the application. Trying to combine all services into one monolithic application will potentially compromise systems stability because if one service crashes the whole application will not be available for use. As a primary language for building the backend of the application Golang was chosen as it is lightweight, allows concurrent code execution and overall well tested by the industry giants such as Google, Netflix, Uber etc. Below are how we would achieve non-functional requirements of our application such as performance, reliability and security.

Performance:

- Response time: We intend to create effective caching mechanisms using Redis which is NoSQL, in-memory key-value store. In particular, we intend to use its fork created by Snapchat KeyDB which unlike Redis supports multithreading thereby increasing speed. Moreover, Our microservice architecture will utilize the gRPC framework to its best as it uses HTTP 2 which supports the streaming of requests and does not wait for a response to send the next request as HTTP 1.1
- Scalability: By dividing our application into smaller modules each with its database and server we are open to horizontal scaling. This means potentially we can add infinite nodes to our services to increase its capacity to handle more

traffic as we scale. It also allows us to scale each service independently from one another

Reliability:

- Availability: We use Kubernetes which is a container orchestration system created by Google. It is mostly used by companies because of its self-healing capabilities and high availability. Kubernetes clusters continuously check nods' health and automatically restart in case of failure
- Fault tolerance: Kubernetes also can perform system automatic rollbacks if something goes wrong.
- In the case of the database, we are using synchronous master and slave nodes which is common practice in the production of data-intensive applications. Basically, all the changes done to the database will be first done on the master and replicated to its slaves. In case the master node is down or some malfunction occurs our system automatically promotes one slave to the master and the system can function without downtime. This ensures data consistency and reliability in case of database failure. Replication of data can also be used as backup if it is needed.

Security:

- Authentication and Authorization: Mainly it is maintained by using tls certificates across services, using JWT for authentication and using API keys to restrict access to the system to users with malicious intents.
- Logging and managing user activities is an essential part of a system. To monitor user activity and error and info logs of the system Elasticsearch in combination with Kibana is used. By defining specific Regex we are able to extract metrics and specific error messages. This enables us to debug our codes easily and detect errors faster and more effectively.

Design

To build our application we have decided on using microservice architecture. One of its advantages is horizontal scaling which enables independently scale each service and potentially infinitely scale. Moreover, when one of the services fails it will not affect the states of the other microservices.

Our application will have several microservices that will run in the backend:

1. Authentication
2. Event Management
3. Chat
4. Recommendation
5. Ticket Management
6. Map
7. Notification
8. Log & Monitoring

Microservice architecture enables us to utilize flexibility so that we can take advantage of the team members and potentially write each service with different programming languages that suit the purpose it. For example, we can use Python to build recommendation models and user Java Spring framework to build user authentication and authorization as it provides sophisticated control via Spring Security framework which eases the development process as it handles most of the repetitive tasks.

Nevertheless, microservice architecture has its own disadvantages. First of all, is data transmission and serialization which could potentially introduce overhead and slow the whole system down. To solve this problem we are planning to use the gRPC framework instead of traditional RESTful services. gRPC is a cross-platform open source high-performance remote procedure call framework. It serialized data to binary and uses HTTP2 which is vastly superior in speed to conventional HTTP 1.1 used in RESTful services. We took proof from LinkedIn engineers who posted that using gRPC reduced latency by 60%(LinkedIn, 2023).

Below we present basic interfaces to be implemented in order to build our application. Each box represents a microservice and it has basic interface functions to be implemented. At first glance it can be seen as simple but here is the only function that can be accessible via external API.

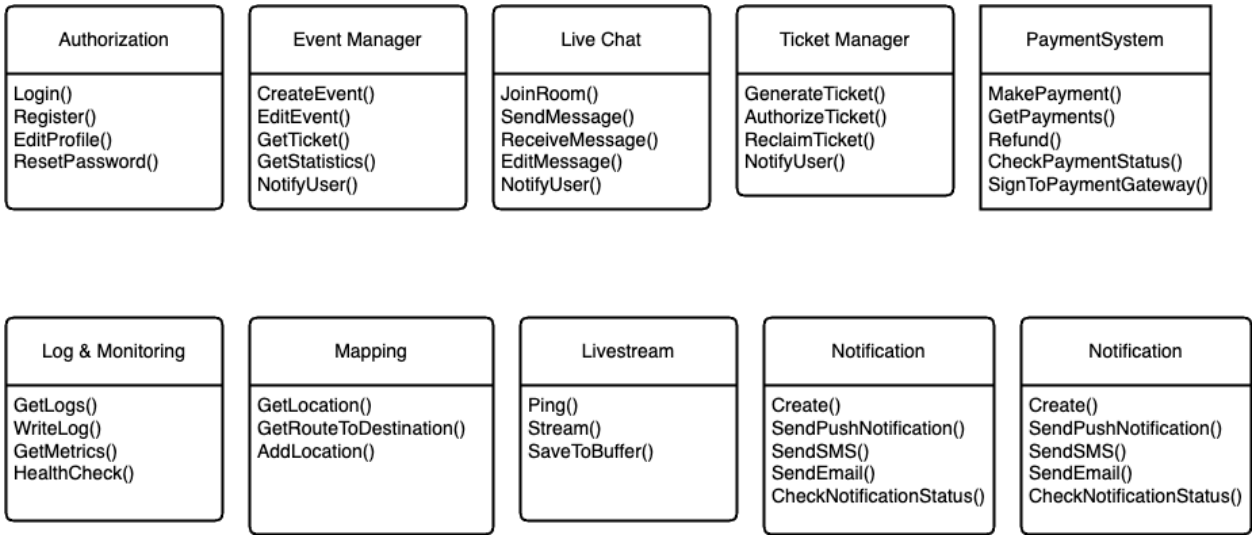

Figure 4. Base microservice functions

The Image below represents how services interact with each other and which services.

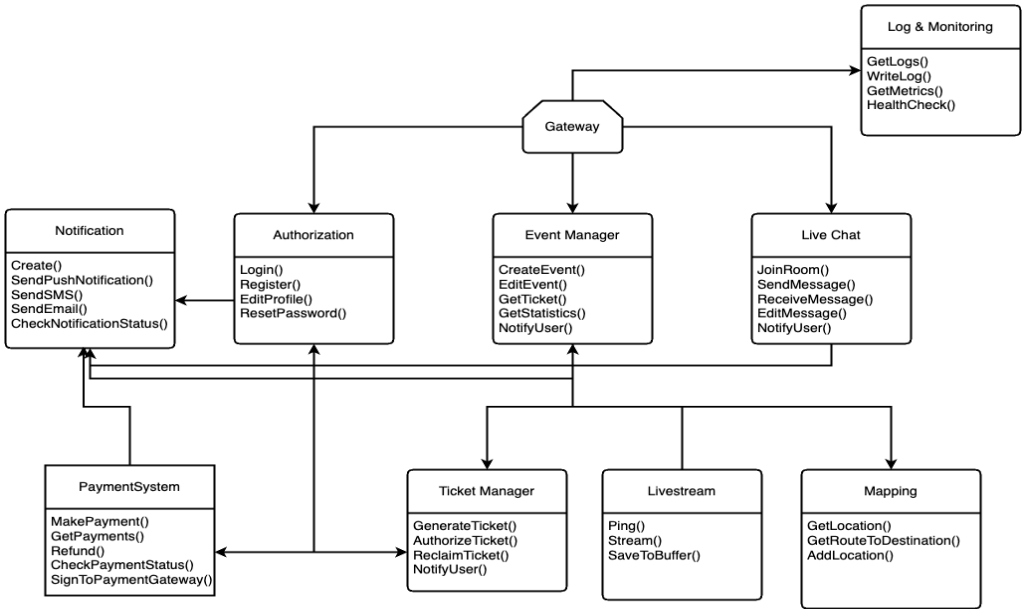

Figure 5. Service Interactions

In the case of data storage, we are using Master-Slave database replication alongside with Cache sharding. All write operations (inserts, updates, deletes) are performed on the master database. These changes are synchronously replicated to the slave node. We chose synchronous mode even

if it is a little slow because it is extremely complicated to handle asynchronous replication without breaking the consistency of the data.

Master-Slave Database Replication: The Database contains several nodes one of which will be the master node on which all the operations are performed and results are replicated on slave nodes. Slave databases are read-only; read operations are distributed among the slaves. Traffic on the master-slave is distributed among them ensuring better overall performance.

Requesting data from the database is costly and time-consuming. To avoid it, information that is frequently accessed can be stored in a cache. It reduces response time but it also can be subject to high traffic which potentially will slow the system down. To avoid it technique called Sharding is used. Sharding is the distribution of data into different nodes thereby partitioning the database. By distributing load it enables parallel access which dramatically reduces response time.

By utilizing this technique we achieved high availability and scalability as if traffic increases introduction of a new partition and slave node is enough. However, it introduces complexity in terms of managing replication, dealing with eventual consistency, and ensuring proper synchronization between the master and slave databases.

Use case Diagram

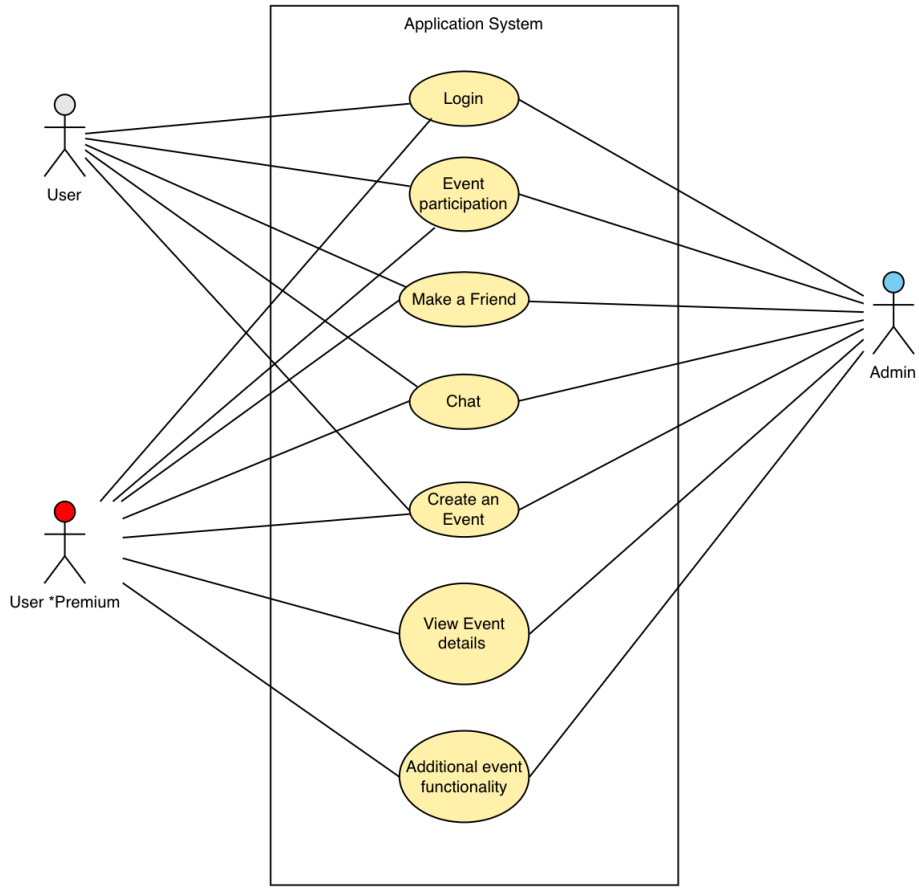

Figure 6. Features access control

Sequence Diagrams

This section contains sequence diagrams of the service and its functionality.

1. Below is shown sequence diagram for the Authorization service. Particularly Login, Registration, Password Reset and Update profile functionality

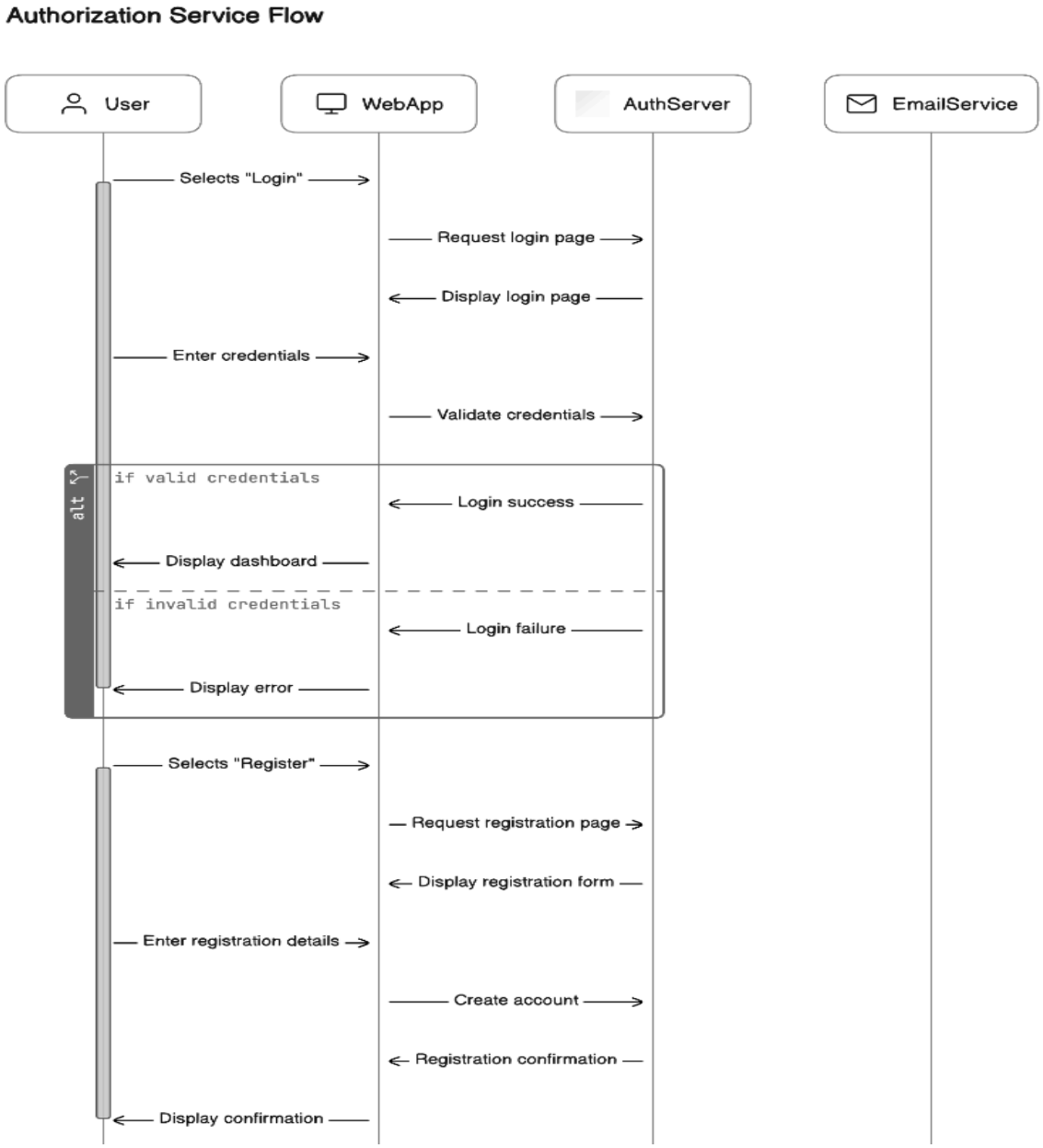

Figure 7. Login and Registration

Authorization Service Flow
User
WebApp
AuthServer
EmailService

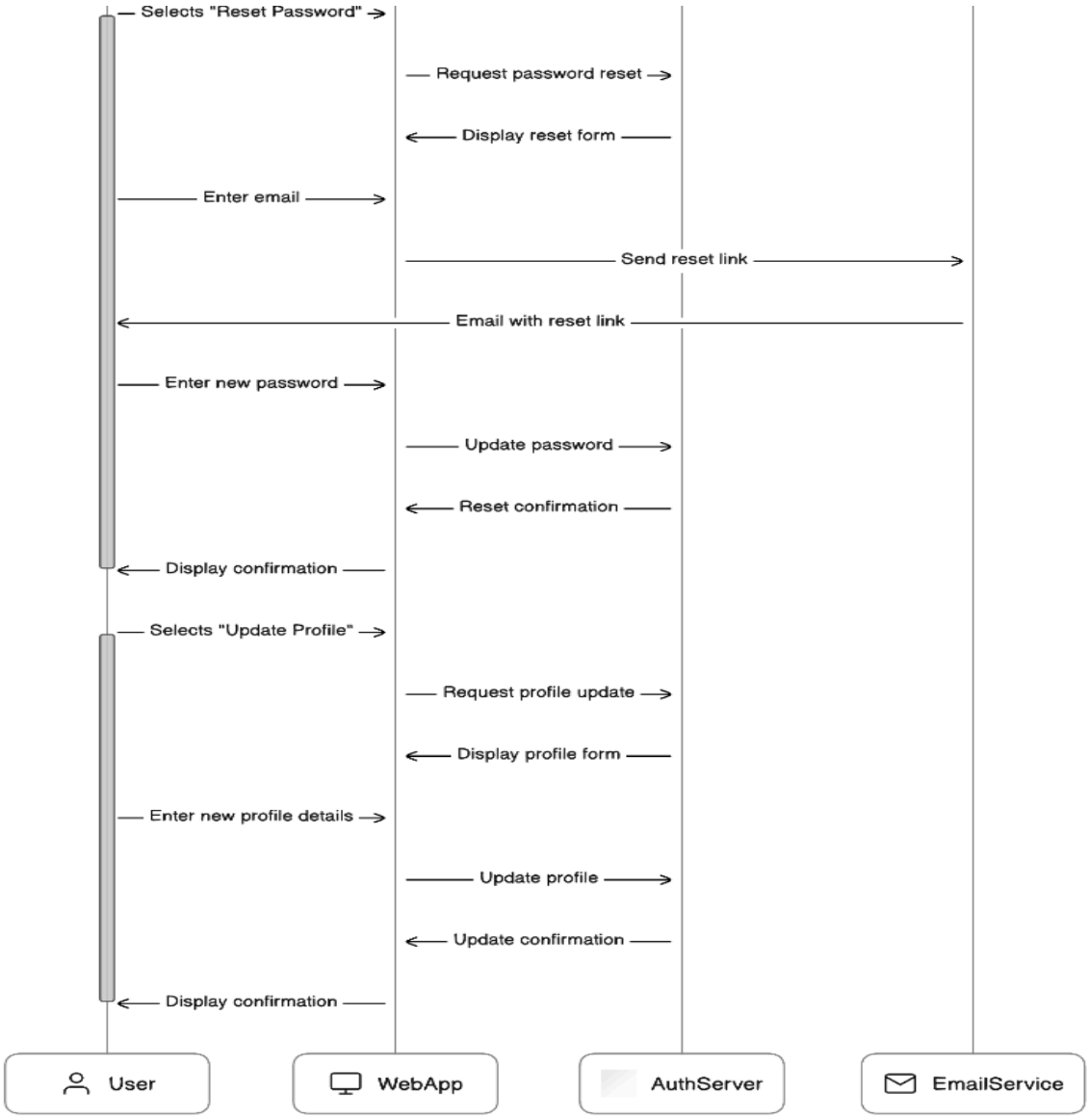

Figure 8. “Reset Password” and “Update Profile”

2. The diagram below shows how process of event creating and hosting. It consists of Authorization, Event Management, Mapping, and Notification services

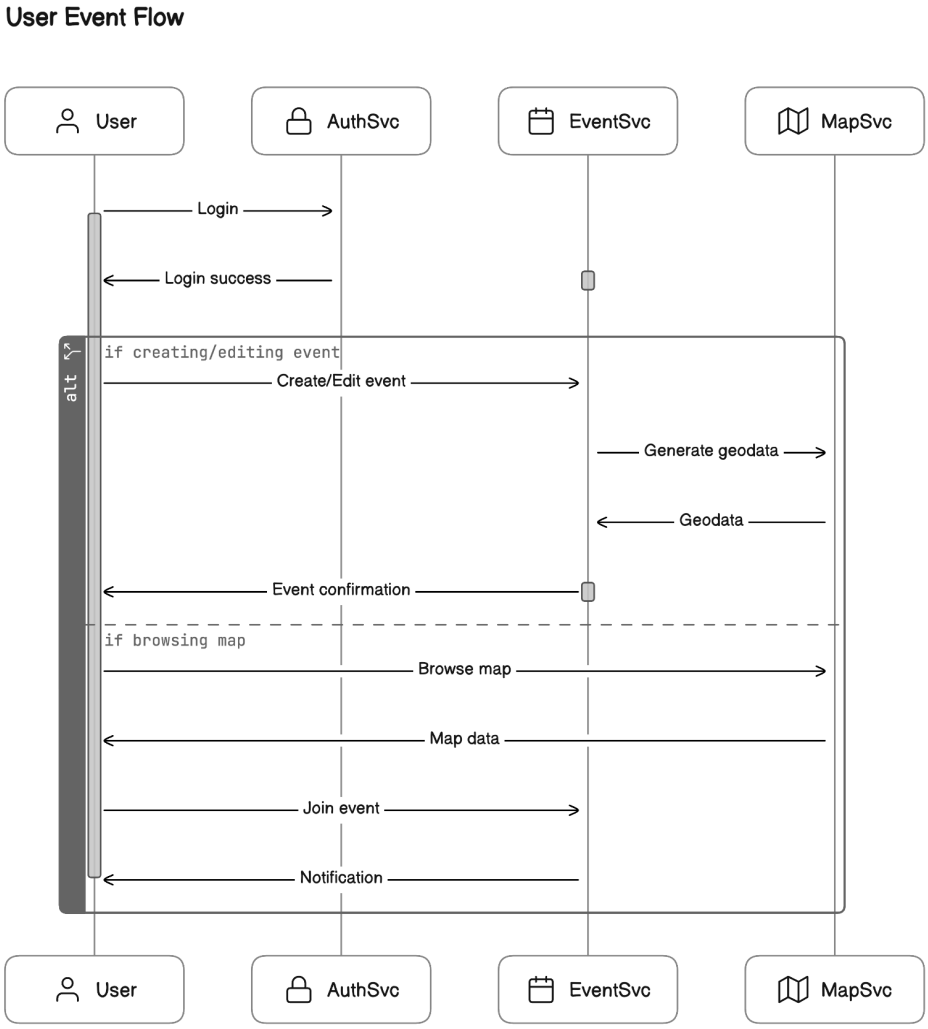

Figure 9. Joining event

3. The diagram below represents the chatting service sequence diagram. It involves Event management, Chat, and Notification services.

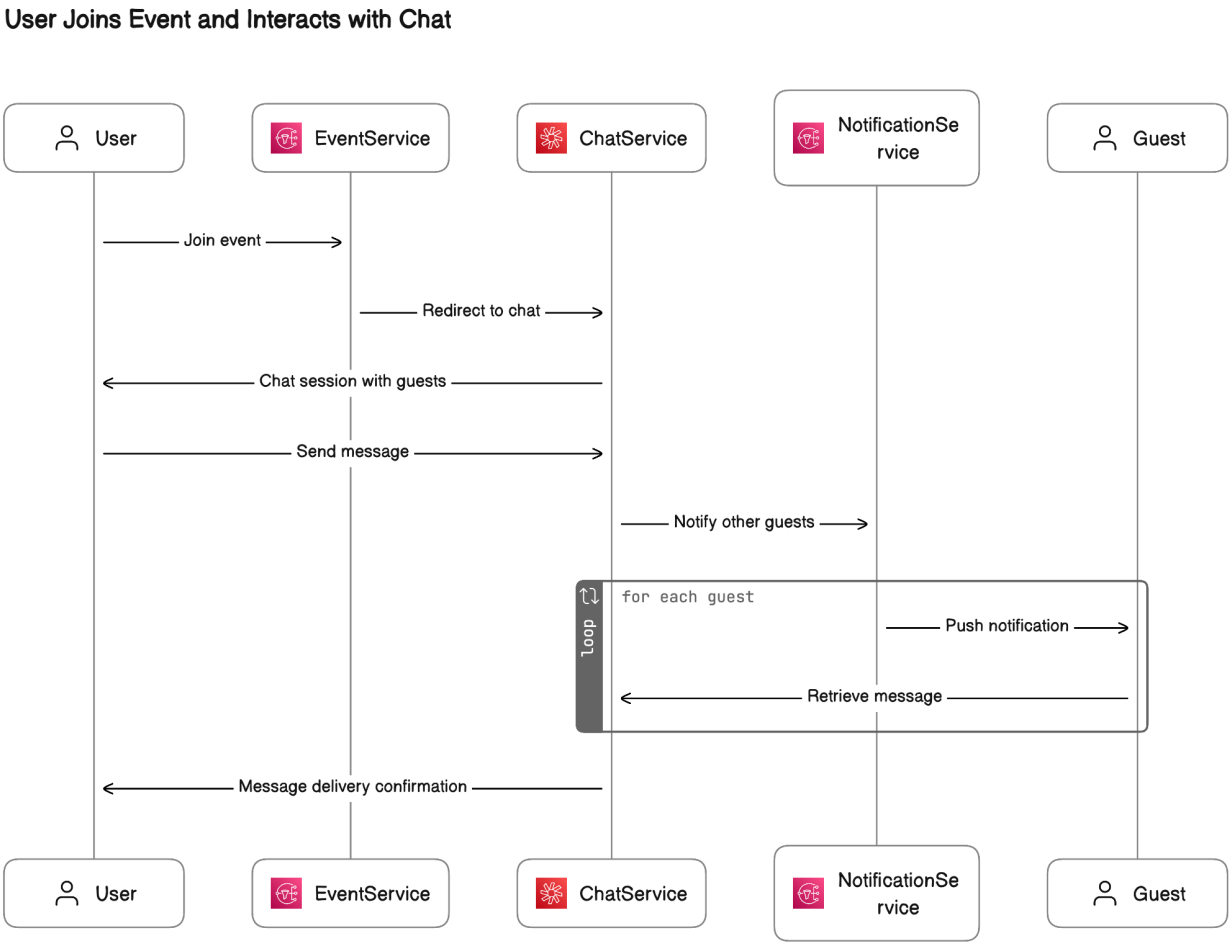

Figure 10. Live chat

Entity Relationship Diagram of our system is shown below

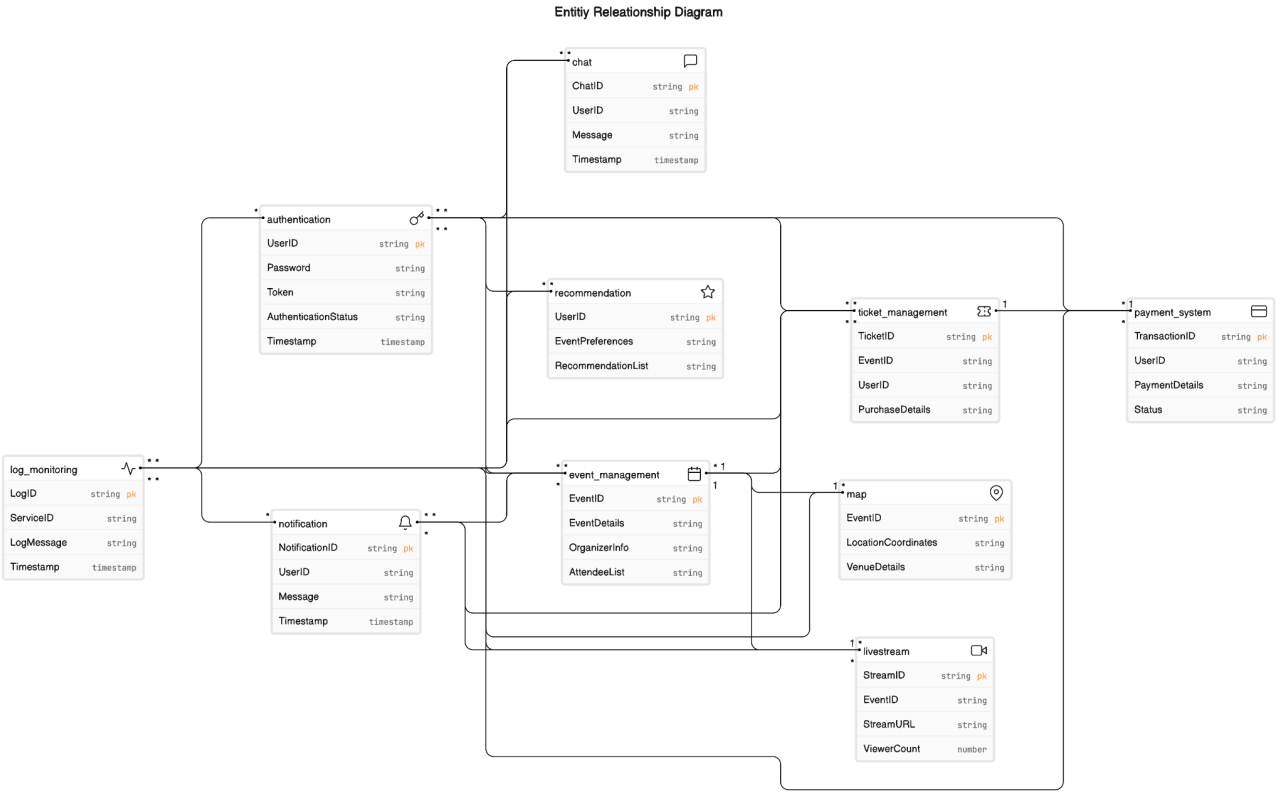

Figure 11. ER diagram

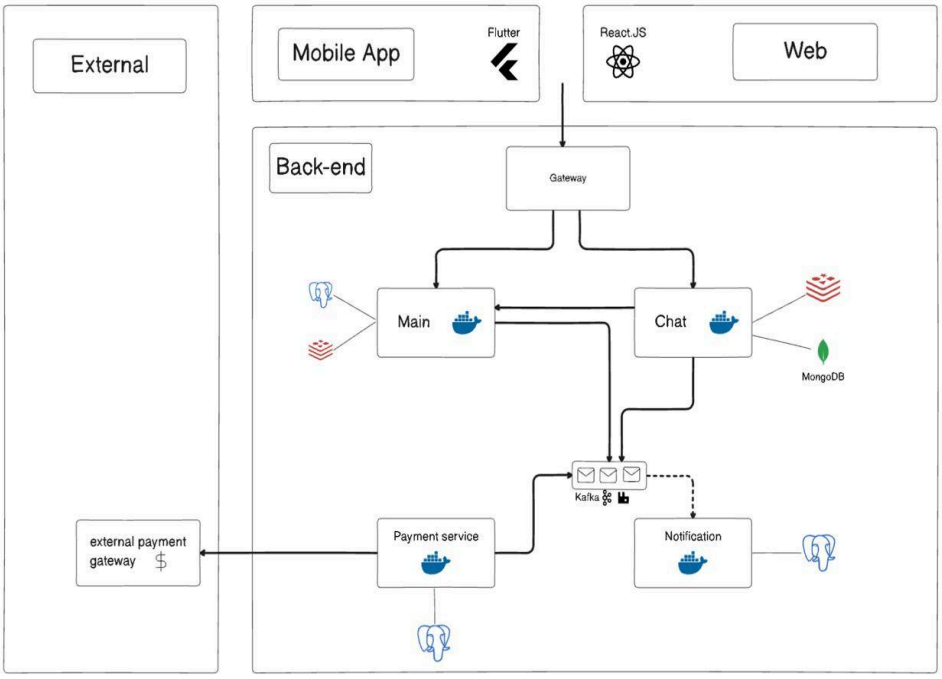

Figure 12. Architectural model

From the architectural model above, we can see the relationship between two clients(in our case Web Client and mobile application) and our event server side. Every request from the client is handled by the gateway which distributes it accordingly. Every service can send requests to other services so that every service is connected, and they talk to each other. All activities are sent to the notification service. Looking closely at the payment service, our server has an external connection with an external payment gateway that will handle all payment requests. Every service has its own database so there is no need to send it to other places or services.

Project's stack:

Software Engineering methodology: During the process of creation and development of our application we used agile methodology, since this methodology was the most suitable for us. We had to frequently show and inform the professor about our current process and ask for improvement of certain things and functionality.

1) Mobile application:

Development tools:

Flutter SDK: Flutter SDK is used to write code in Dart programming language, compile and debug the program

Dart programming language: Flutter code is written in Dart programming language

Android Studio is used for developing the program

GitHub is used for collaborating with groupmates during the project.

Third-party tools:

The HTTP library is used in order to fetch data from servers and make HTTP requests.

OpenStreet Map is used as map API for application

2) Frontend(Web application):

Development tools:

React.JS is a JavaScript library used to create user interfaces.

Next.JS is a framework for the **React.JS** library that adds additional functionality and makes some things easier to implement like Routing between pages etc.

Third-party tools:

Due to the fact that our users should be able to observe and check events on the map, it required us to integrate YandexMap into our project, therefore YandexMap library was implemented and imported to show the map and events.

Sass (Syntactically Awesome Style Sheets) was also imported to the project, which is a preprocessor scripting language that is interpreted or compiled into Cascading Style Sheets, which makes styling easier and more convenient.

3) Backend:

Development tools:

- *Golang:* Go is an open source programming language that makes it simple to build secure, scalable systems. It has all the tools to create scalable and robust applications. Using this we build several microservices that communicate with each other via gRPC framework
- *PostgreSQL:* relational database management system. It is ACID-complient meaning it provides fault tolerance to our application. Primary storage medium of our application.

- *Docker:* allow to separate applications from your infrastructure allowing to ship your application on any device. This process is called containerization.

Third-party tools:

- *Firebase Cloud Messaging:* is a cross-platform messaging solution that lets us send messages to devices. It was used to deliver push notifications, notifying users of recent updates in events, chat messages and
- *Elasticsearch and Kibana:* user for log monitoring as it allows to search, categorize and visualize logs produced by the application
- *AWS S3:* object storage built by Amazon. Primarily used to store images files related to events, guests and hosts.

Project Execution

At the beginning of the Fall 2023 semester, we decided to conduct a survey for future project implementation. The survey's results clearly showed the actuality of the problem connected with the event management system in our university and allowed us to identify the main features that we must implement in our project.

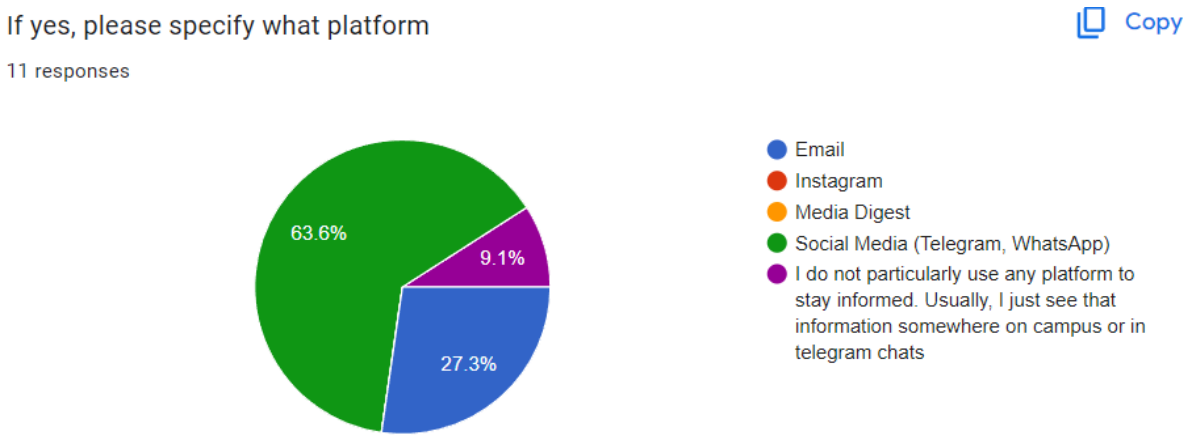

Figure 13. Survey results: Currently used platform

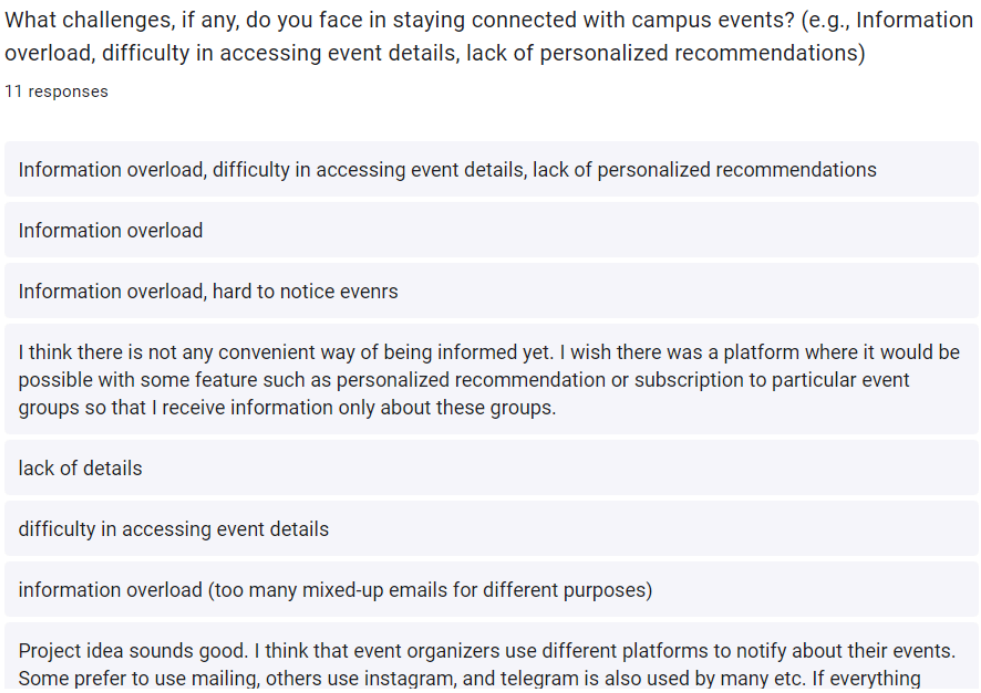

Figure 14. Survey results: Current challenges

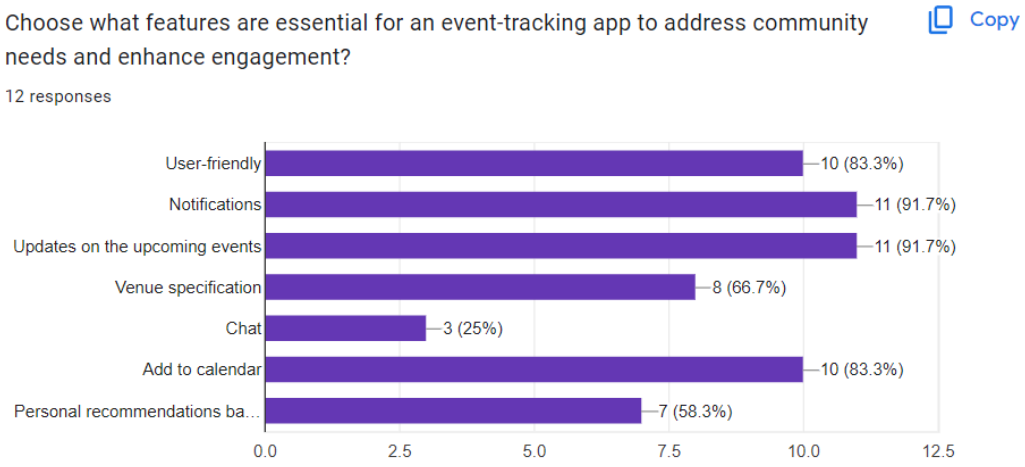

Figure 15. Survey results: Essential features of the project

Based on the results we identified that there is a demand for our application not only from students but from student organizations as well. The primary functionality of the application centered around hosting and attending events. Cross-platform approach was also welcomed by respondents.

For the prototype of the application at the beginning monolithic architecture was chosen, however during development we have noticed that we would provide various independent

services, hence moving towards microservice architecture was researched. Although monolithic applications are simple in their design, eventually we decided to stick with microservice architecture as it allowed us more flexibility and collaboration. Because workload and functionality can be divided into small services, it enforced better collaboration as there is less functionality to implement per service. “Divide and conquer” strategy was great for us as working on pairs of small tasks accelerated the development process. Apart from the development process, after reviewing several conference papers regarding microservice architecture written by De Lauretis (2019), Rademacher (2018) and Dinh-Tuan (2020). This paper came to one conclusion that distributor services are difficult to manage in enterprise scale applications because of many moving parts and complexity. However, authors also suggested using micro services if the application has many independent parts, and has to scale in a long run.

Integrating Virtual Map was suggested to improve user experience. This posed some challenges as reliable and consistent web mapping applications are rare and often cost money to operate. For this reason we decided to utilize 2 different mapping APIs commercial and open-source: Yandex Map and OpenStreetMap respectively. Major challenge of working with a map was to interact with it, meaning placing markers, converting longitude and latitude data into human readable addresses, searching for addresses and much more because of lack of proper documentation. However, with the help of the open-source community we have overcome this problem.

One technically challenging problem was to deploy all the microservices on to cloud providers as every service is interconnected with each other we had to ensure seamless connectivity as well as fault tolerance in case one of them is not available. To ensure this we used Docker containers with Kubernetes container orchestrator. Docker enables us to detach from our own machine infrastructure and allows us to choose a suitable operating system and customize it with only tools required for our application without having to install unnecessary apps that could potentially harm the performance of the main application. After creating containers we needed an autonomous system that would monitor and orchestrate all of them. Meaning rolling back to previous versions in case of error, automatic resource allocation and many more. After trying to deploy the Kubernetes cluster we noticed that it is extremely difficult to build from scratch with zero to none knowledge in DevOps. Configuration and deployment was to some extent difficult

to sustain in our team as we previously have not had an experience with Kubernetes. Therefore we have decided to just use Docker containers with a compose option.

Managing persistent connections is always hard and was another challenge that came across while developing the live chat service. Keeping all these connections using Golang posed a challenge of running out of CPU memory and the kernel just shutting down all the running processes. After investigation CPU memory usage bursts was the reason for leaking goroutines which were not properly terminated and were taking up CPU memory as they were infinitely trying to establish connections with clients. After careful examination of synchronization methods used in the code the weak spots and fault inducing scenarios were examined and solved.

Cross platform development of mobile applications was challenging because of the not suitable equipment for IOS development. We initially planned to do a cross platform mobile application using Flutter using which one can write code once and compile it for different systems. The Flutter developer had a Windows system installed in his local machine which enabled him to build apps for Android devices. However, when time came to IOS we had a problem because none of us had MacOS to recompile code for the IOS. Eventually we found a device running on MacOS to compile our application for IOS but still encountered problems with compilation because of XCode which is Apples' development environment which often causes unexpected errors and overall development process is slowed down because of slow compilation on XCode for Flutter application and spontaneous unexpected errors. Having little experience working with MacOS and IOS overall we have postponed our plans to release an IOS application and concentrated our efforts on Android for the duration of the course.

Recommendations system was the hardest part of the project as none of us are experienced in machine learning or even python. Collaborative filtering was chosen as the main approach to get recommendations for the users. Collaborative filtering is recommending suggestions based on other users' similar activity which utilizes matrix factorization dividing 2 tables of features, users and events. Model using these factorized tables tries to predict the whole table where the entries represent the score of how likely the user will use it. Drawbacks of this algorithm is that it is offline training meaning that if users' taste changes algorithms will not automatically suggest the places to visit it will suggest cached places but the next day a person will get new suggestions

based on changes in his taste. This limitation is taken into consideration and will be fixed in the future.

Project Evaluation

With the aim of proper project evaluation, we conducted one more survey at the end of our project work. We showed our project to different students of our university and asked them to use and check all the project's features and give feedback at the end. Firstly, we wanted to evaluate the visual part of the project and asked respondents to give a rating for the visual part of the project where 1 - Lowest mark and 5 - Highest mark. Almost 60% of the respondents gave the highest mark for the visual part. 25% of respondents rated as 4 and only 1 respondent rated as 3. It is a satisfactory statistic for us because almost all respondents gave positive marks for the visual part.

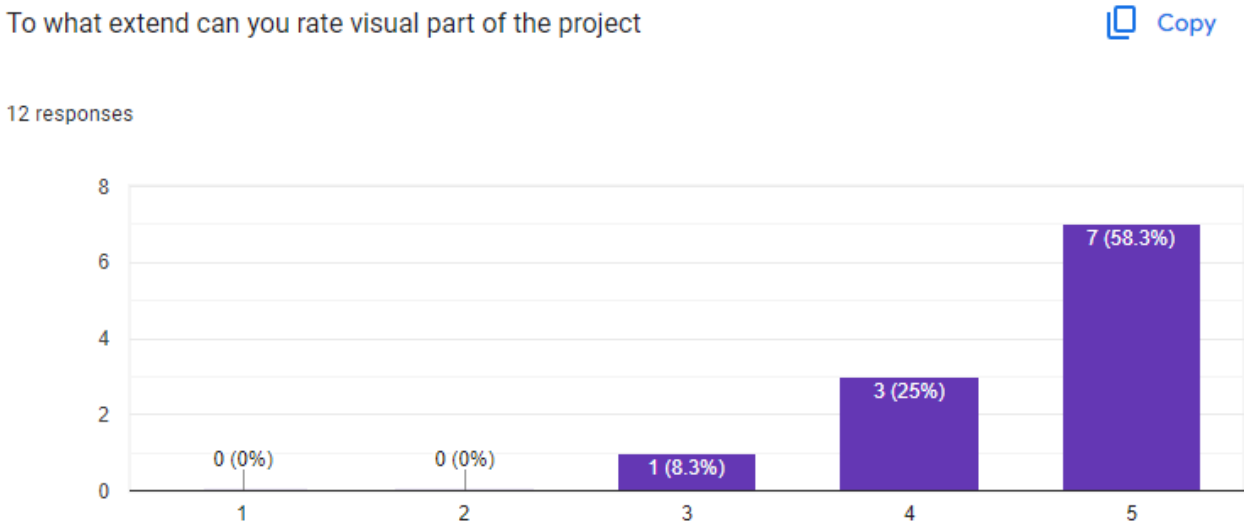

Figure 37. Survey result: visual part

After rating the technical part of the project we asked our main question to determine whether it did or did not solve the problem connected with the events organization in our university. 8 out of 12 respondents stated that our project "Fully decides" the problem. These statistics clearly show that we managed to find a right solution to the identified problem.

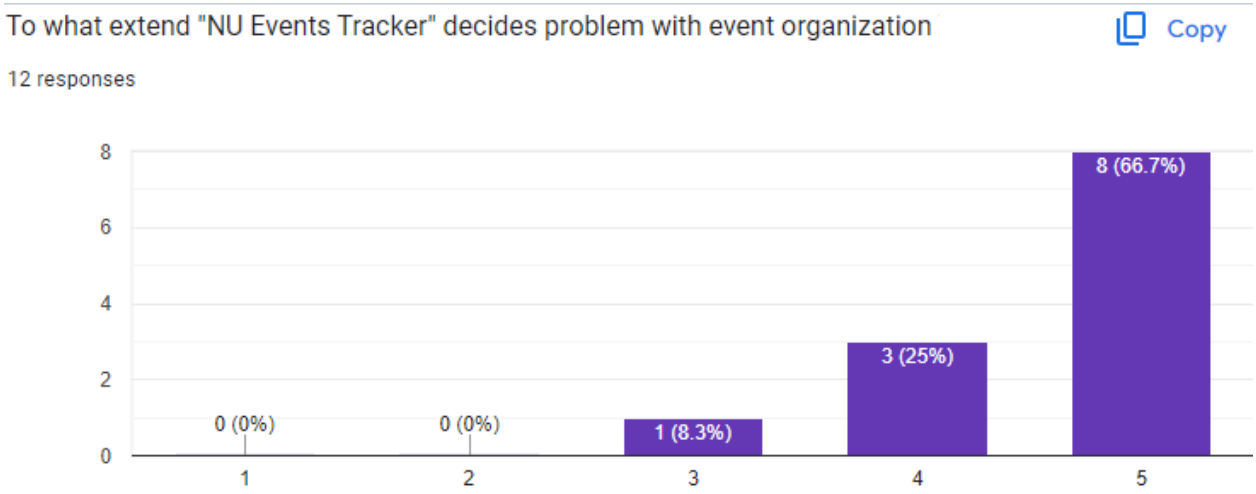

Figure 38. Survey result: Main evaluation

Finally, we asked respondents about the future use of the application. All of the respondents stated that they will use our project in the future which opens a great opportunity to expand the current system and scale it in the future.

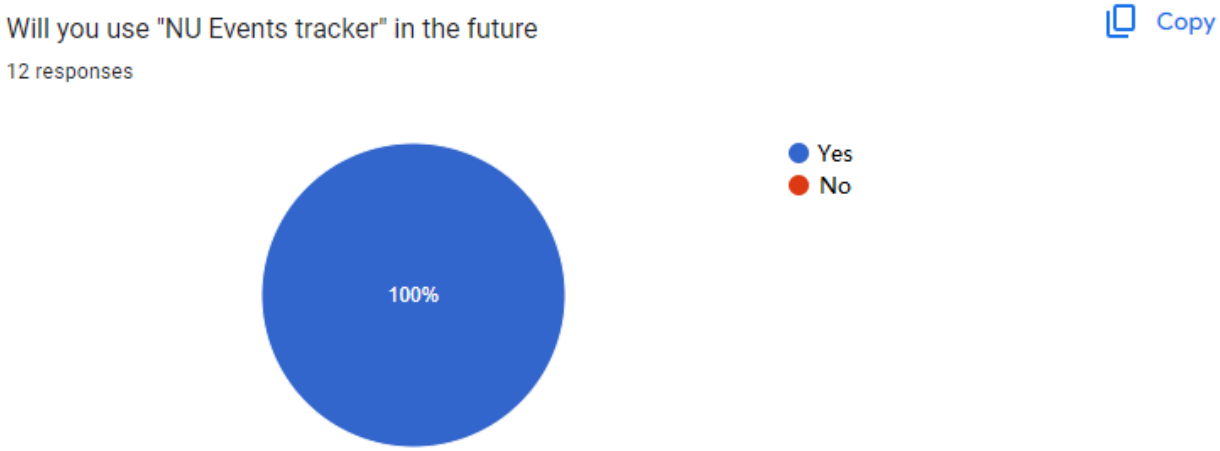

Figure 39. Survey result: Actuality of the project

Technical Evaluation

When it comes to the technical part of the project, it can vastly improve further based on the already established code. Technologies used in the project are modern and yet tested with time and surely will be relevant even after several years.

Conclusion

In conclusion, the development and implementation of the "Nazarbayev University Events Tracker" system marks a significant step forward in addressing the challenges associated with event organization within our university. Through analysis, design, and implementation phases, our team has successfully created a comprehensive solution that streamlines the management of university events. As we continue to refine and improve upon this foundation, we are confident that the system will serve as a valuable tool for advancing our university's mission and objectives.

We identified possible ways to improve our project in the future:

- Live streaming. Organizers will have the opportunity to make a live streaming of their events right inside our application for users who couldn't attend the event offline.
- Scaling to the city level. There is a big opportunity to scale our project to the city or even country level, which can create amazing opportunities which help people around the city of Astana to create and promote their events.
- Offering other universities to integrate our system. We could collaborate with different universities across the country and offer them our system. Since all the universities will adopt this application in their systems, it will increase recognition of our university, due to the fact that this application was developed by NU students.

References

- Ghofrani, J., & Lübke, D. (2018). Challenges of Microservices Architecture: A survey on the state of the practice. ZEUS, 1–8.
- Hinkula, J. (2018). Hands-on full-stack development with Spring Boot 2.0 and React: Build modern and scalable full-stack applications using the Java-based Spring Framework 5.0 and React. Packt Publishing.
- De Lauretis, L. (2019). From Monolithic Architecture to Microservices Architecture. In 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) (pp. 93-96). Berlin, Germany. DOI: 10.1109/ISSREW.2019.00050.
- Rademacher, F., Sorgalla, J., & Sachweh, S. (2018). Challenges of Domain-Driven Microservice Design: A Model-Driven Perspective. IEEE Software, 35(3), 36-43. DOI: 10.1109/MS.2018.2141028.
- Dinh-Tuan, H., Mora-Martinez, M., Beierle, F., & Garzon, S. R. (2020). Development frameworks for Microservice-based applications. Proceedings of the 2020 European Symposium on Software Engineering. DOI: 10.1145/3393822.3432339
- Khuat, T. (2018). Developing a front-end application using ReactJS and Redux.
- Lazuardy, M. F. S., & Anggraini, D. (2022). Modern front-end web architectures with React.js and Next.js. Research Journal of Advanced Engineering and Science, 7(1), 132-141.
- Ridwan, M. (2015, February 4). Predicting likes: Inside a simple recommendation engine's algorithms. Toptal Engineering Blog. Retrieved from https://www.toptal.com/algorithms/predicting-likes-inside-a-simple-recommendation-engine
- Pistocop. (2022). How a recommendation system Web App was Built. HackerNoon. Retrieved from https://hackernoon.com/how-a-recommendation-system-web-app-was-built
- LinkedIn integrates protocol buffers with Rest.li for improved microservices performance. (n.d.). LinkedIn. https://engineering.linkedin.com/blog/2023/linkedin-integrates-protocol-buffers-with-rest-li-for-improved-m