Computer Science Department

Senior Project Final Report

Title of the project:	“Multi-Organization Pass System for Children’s Extracurricular Activities"
Team Members:	Abrorbek Shanazarov Aniyar Durmagambetova Marzhan Saparbekova Samat Sauranbek Ilyas Kudaibergenov
Project Advisor	Khalil Khan

Executive Summary

This project addresses the issue of accessibility and efficiency faced by parents and children in Kazakhstan's extracurricular education. The primary objective of this project is to design a centralized platform that enables parents and their children to explore, register and manage multiple extracurricular activities through a single mobile application and single monthly payment. The project also empowers activity providers to offer their services in a more effective way. In order to enhance personalization and usability, the system incorporates machine learning modules that provide content recommendations and schedule optimization based on user behavior.

The delivered product was built using SwiftUI with Apollo GraphQL, implemented in Java with Spring Boot, supported by PostgreSQL, MinIO, Keycloak, and Temporal.io. The ML component offers endpoints for real-time personalized recommendations and automated retraining.

Introduction

Problem

For parents, finding and managing extracurricular activities for their children can be a time-consuming and overwhelming experience, especially when the children are just starting to explore their interests and needs. In Kazakhstan, organizations for extracurricular activities primarily use social media to offer their services and require separate third-party apps to manage the registration, payment and attendance processes. This lack of centralized system results in ineffective processes carried out on both parents' and organizations' ends.

Motivation and significance

This project was designed as a solution to this problem. By offering a unified platform to carry out all the processes required in registering for extracurricular activities, this project aims to simplify both parents and service providers, as well as empower children to explore a wider range of activities that can contribute to their education and well-being. Improving the user experience of accessing activities can make a significant impact on giving well-rounded children education.

Solution overview

The proposed solution is a centralized mobile application that allows parents to make a singular monthly payment that gives them access to browse and register for numerous activities. The system also includes additional AI-powered tools that enhance activity exploration based on personalized suggestions and schedule management to avoid conflicts.

Outline

Problem identification and solutions exploration was presented in Background and Related Work section. System design deliverables can be viewed in Project Approach section. Technical implementation details are described in Project Execution part of the report. The system was evaluated through user testing which is described in Evaluation section.

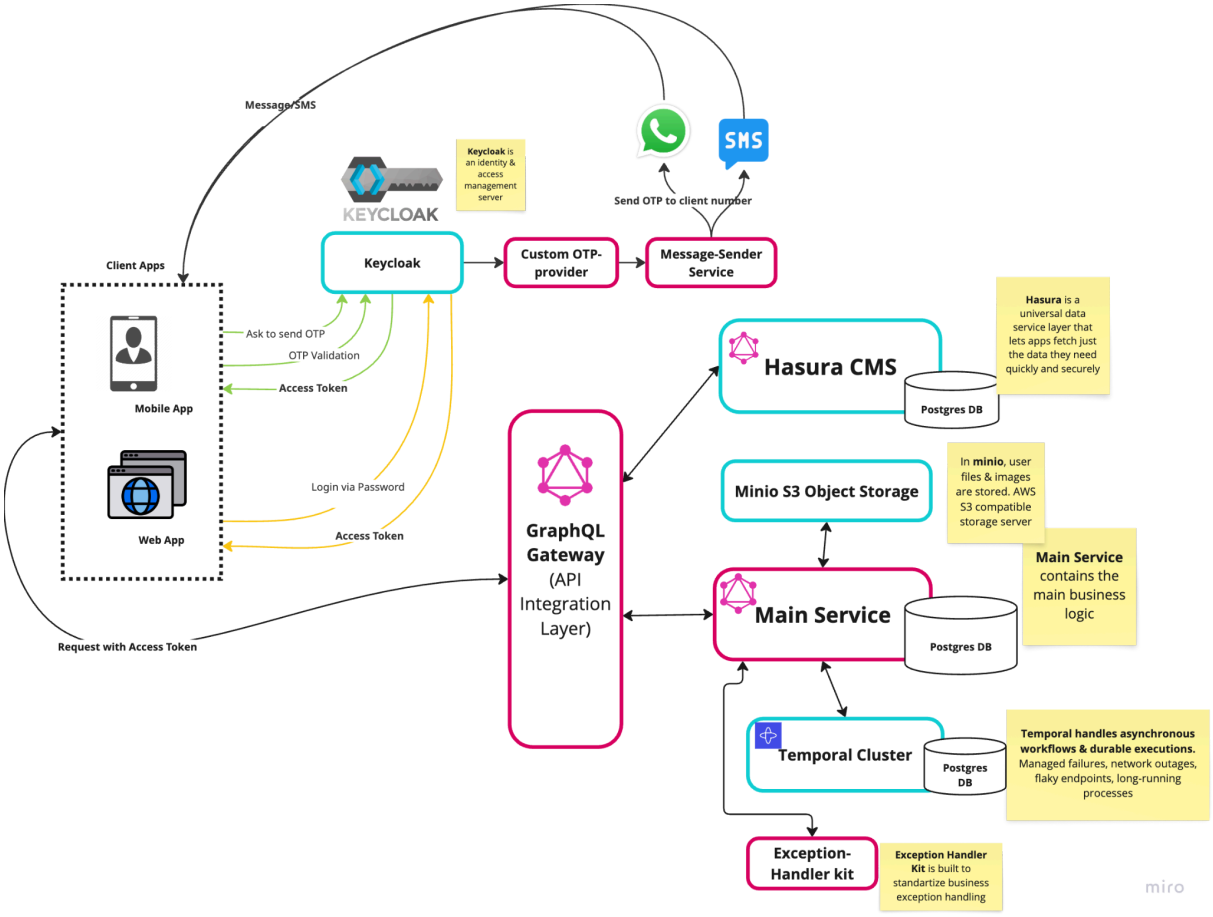
Background and Related Work

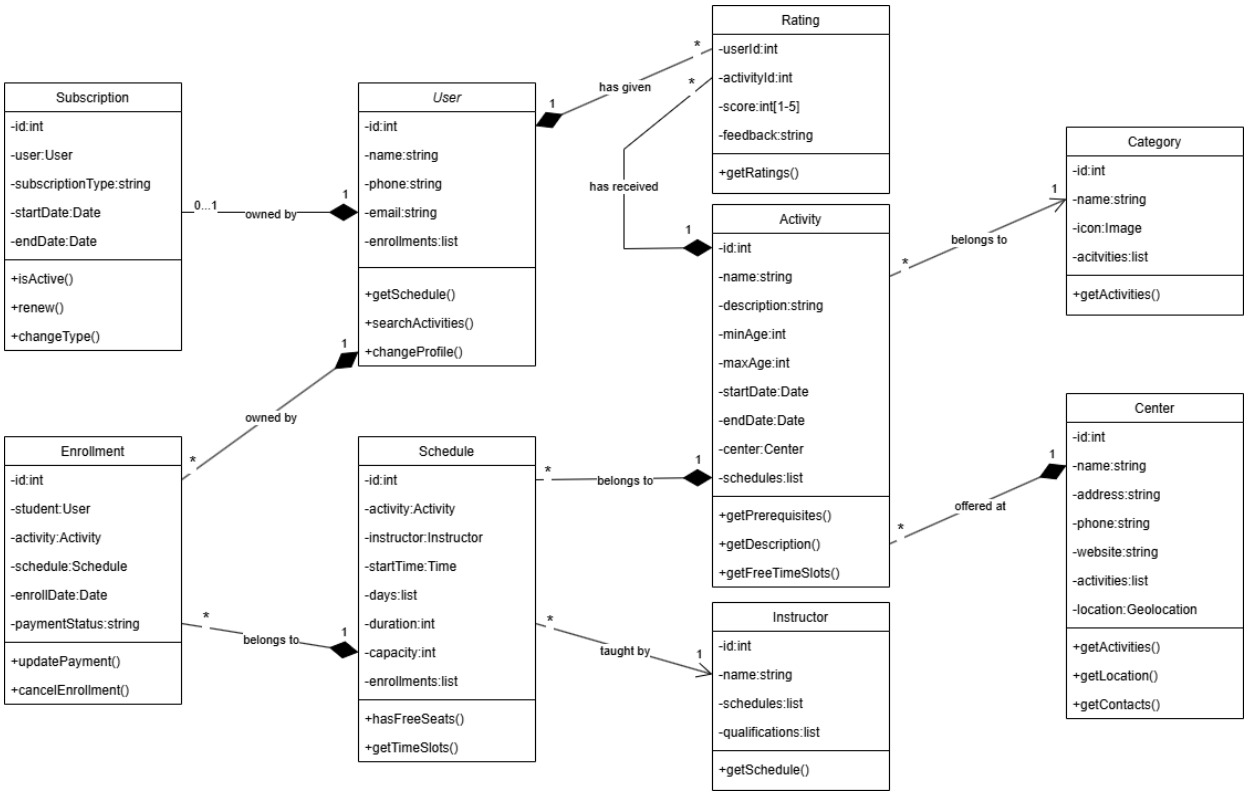
A widely known sports access app, 1Fit, inspired this project. It also operates based on a single-pass system that allows adult users to visit multiple sports activities. However, the scope of that project is limited to sports classes and gyms, while we aim to enable children to access a wide variety of extracurricular activities, from robotics classes to ballet rehearsals. Another inconvenience of 1Fit is its lack of deep personalization. It does not include any additional tools to make the browsing process more effective by suggesting new activities based on previous registrations. Moreover, there is no schedule optimization system that would allow to avoid time conflicts or consider the user's regular schedule. The idea for this project was to adopt existing unified pass systems and enhance it with the use of machine learning and an engaging user experience.

Personalized machine learning recommendation systems have become a standard tool for effective user experiences. Numerous cutting-edge ed-tech platforms have incorporated content-based recommendation systems that use history or engagement data to adapt the suggestions of the feed to the highest possible accuracy. Coursera reports that implementing an ML suggestion system has significantly improved course enrollment and completion rates (Mercado, 2025).

Project Approach

Software architecture





Backend: The platform was built with Java 17 using Spring Boot 3, following a modular microservice architecture deployed on DigitalOcean. For API interactions, it uses GraphQL through a unified graphql-gateway for aggregating schemas and managing authorization. Persistent data is stored in PostgreSQL and MinIO provides S3-compatible object storage for images users uploaded. Authentication is handled by Keycloak, enhanced with a custom OTP provider that uses WhatsApp and SMS via the message-sender microservice. Background jobs or recurring tasks such as weekly activity slot generation are handled using Temporal.io. Translations are managed with Hasura which enables real-time multilingual content via GraphQL. Firebase Cloud Messaging allows real-time push notifications, and all user roles, from administrators to parents, interact with the system through role-based flows including QR-based attendance check-ins.

Frontend: The iOS mobile application was built with SwiftUI. It also implements Lottie for smooth, vector-based animations. For backend communication, the app uses Apollo GraphQL client for type-safe API interactions that ensure minimal network overhead. The architecture follows MVVM pattern with SwiftData for local persistence. User authentication is handled through secure token-based flows, and the QR scanner module utilizes AVFoundation to enable attendance verification at activities.

Roles and features

Parent: Registers, manages children's accounts, views and books activities, and makes payments

Child: Views assigned activities, marks attendance

Organization admin: Creates and manages activities, checks attendance

Workflows

User registration and onboarding

- Parent installs the app
- Signs up using their email, phone number and verifies their data in WhatsApp
- Registers the child account
- Enters age, interests, location and regular schedule
- Lands on dashboard

Activity browsing

- Parent is redirected to “recommended” section of activity feed
- Browses the feed, changes the filters (age group, type, proximity, time of day), if needed
- Selects an activity and views details

Activity registration

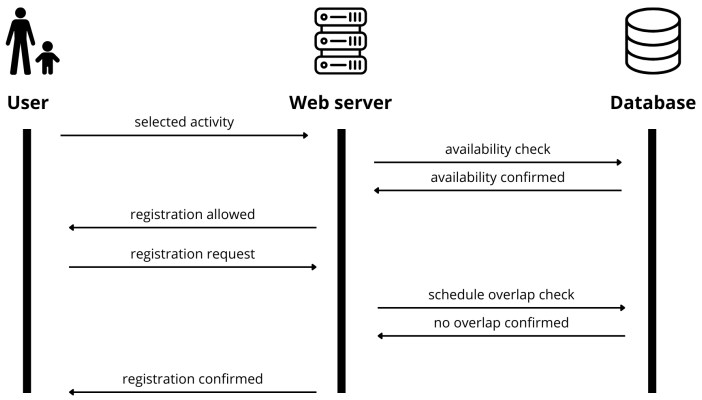
- Clicks “register”
- Receives a message about schedule compatibility
- Receives registration confirmation

Attendance tracking

- Both parent and child receive a notification about upcoming class
- Child scans a QR code on the entrance to the class
- The attendance is marked

Organization activity posting

- The organization administrator signs up to the admin page
- Creates an activity listing, provides details
- Publishes the listing
- Browses the list of the registered students



Algorithms

The machine learning modules use libraries such as PyTorch, FastAPI. Pytorch was used to create and train the models. FastAPI powered the API for schedule optimization. The models function on a basis of neural network estimating matrix factorization using learnable embeddings. Each user and item ID is mapped to an embedding and used to predict a rating. Model uses a fusion of generalized matrix factorization and multi-layer perceptron to produce an output. The first one captures the linear relations using cosine similarity and the second one captures non-linear relations through hidden layers with non-linear activation. To get recommendations, a user ID and k parameter is passed as an input to the model. Model estimates the user's rating to each item and ranks them accordingly. Then, k item IDs with the highest predicted rating are returned. The API has 3 endpoints: retrain, recommend, and status. Status request returns the last time the model was updated (retrained). Recommend requests accept user ID and k as parameters and return a list of item IDs. Retrain request initializes refetch of the data from a database and retraining of a model. By default, retraining is performed regularly to adapt for new user data. To make the model always available to serve requests (e.g. during training), two copies of the model are stored, one only for inference and the other for training. Unknown user IDs (e.g. new user) can also be passed, then it returns the most popular items.

Tools and third-party components

- **Keycloak:** needed for authentication and authorization. It handles login via username, password and creates secure access tokens. It was integrated with a custom OTP provider to support secure mobile, web access.
- **GraphQL Gateway:** central API integration layer, handles requests from client apps.
- **Custom OTP Provider and Message Sender Service:** handling one-time password delivery.
- **Hasura CMS:** ensures role-based security of data access, powered with PostgreSQL.
- **MinIO:** object storage for user files and media.
- **Main Service:** contains system functionality, interacts with PostgreSQL.

- **Temporal Cluster:** for asynchronous workflows like registration confirmation or scheduling.
- **Exception Handler Kit:** backend error handling.
- **Figma:** design and prototyping
- **GitLab**: version control and project management

Project Execution

Over this academic year, the project has gone through all of the stages of the development cycle. We started with an idea to make extracurricular activities for children more accessible, and are finishing with an MVP product that incorporates features that were initially planned. During our work, we decided to stress clear division of responsibilities in the team. So each member took ownership of specific components in accordance with their best skills and knowledge. Our workflow was agile with daily progress syncs and weekly meetings to discuss plans for the next week and resolve blockers. Backend (Aniyar) and ML engineers (Samat and Abrorbek) worked closely with each other to integrate their parts, while mobile and frontend developers focused on effortless UX. However, all of the key decisions were discussed and made collaboratively.

Fall semester

We started with user research and requirements gathering, and explored children's data regulations and security precautions. We made most of our architectural and design decisions that semester and worked on server-client architecture and implementing core functionalities. Although we initially considered a monolithic backend, later we changed to modular service since we decided to implement a variety of asynchronous tasks that required management. Looking back we see that we made the right decision. Apart from that, during the early stages of backend setup, we faced some challenges in configuring the internal network and firewall to ensure communication between microservices. This issue was significant because our architecture consists of multiple components such as Temporal cluster and GraphQL, as well as external integrations like Keycloak and Firebase. Debugging connectivity issues required some additional time and deeper understanding of connection protocols, and misconfigurations were later resolved, although they caused some delay in our work. Implementing GraphQL was also one of the challenges, as it is a new technology for the team members and required considering not just current use cases, but also future scenarios. There were several query misuses that were further eliminated with the exploration of query optimization techniques.

Spring semester

During the second semester, our focus shifted to machine learning models responsible for personalized recommendations and schedule optimization. Their concept was rethought by our entire team a couple of times as we approached the second semester, but with the feedback from the project adviser, we managed to select the most effective solution to the issue we were

tackling. Although, primarily we did not face any major issues in our work, making a recommender API available and responsive required to consider synchronization and model update issues. Closer to the middle of the semester the setup was integrated into the platform's microservice ecosystem and connected to the central PostgreSQL. While Samat and Abrorbek were working on the modules, Aniyar finalized her work on backend infrastructure, adding final touches to specific domain logic such as scheduling, attendance, notifications and media management. Ilyas while working on the mobile app made a decision to adopt MVVM architecture for better maintainability. And Marzhan built and connected the admin page interface to GraphQL gateway. The last 2 weeks of the semester was devoted to a mini usability testing study that we conducted with our peers.

Evaluation

To evaluate the effectiveness of the delivered product, we conducted a series of user experience usability testing. Our primary focus was on criteria such as usability, efficiency, and satisfaction. The study was conducted among our 8 peers. They were asked to go through 3 use case scenarios: (1) User registration and onboarding, (2) Activity browsing and (3) Activity registration. As a result, all of the 8 participants managed to complete the registration and onboarding process in under 1.3 minutes, which we find to be a satisfactory result. Secondly, 15 out of 15 booking attempts were completed successfully without timetable conflicts at all, or the user being notified about a conflict, which ensured that the schedule optimization system works well with real-life scenarios. Moreover, the system successfully dealt with invalid, incomplete and unexpected data. Edge testing conducted by us also illustrated that the recommendation system verifies all the constraints. As for user satisfaction, the majority of the participants reported that activities recommended by the system comply with their expectations.

Although we understand that the sample size is fairly small, and the evaluation process we applied for this project is more intuitive than technical, it was somewhat sufficient for the core functionalities that we incorporated. We realize the need for more thorough usability testing that would involve more people and larger-sized use case scenarios.

Conclusion and possible future work

Over the course of this year, we successfully developed all of the core functionalities of the system using a microservice architecture that enables parents, children and organizations to interact with extracurricular activities on a centralized platform. Our project incorporated role-based access control, secure authentication, and optimized scheduling and content suggestion. A key contribution of this project is in integration of different technologies like

Temporal.io backend microservices, MinIO data storages, real-time communication through FCM and machine learning models that enhance the usability of the app.

Although we made good progress, still there are areas for improvement. The ML models can be even more functional if more user behavior data and richer context features are incorporated. In addition, a better evaluation techniques should be applied to further improve the system performance by identifying weaknesses that might have been undiscovered. Future work could also include refining the user interface, collecting more detailed feedback from children and mapping that information to recommendation system, as well as including gamification features such as collecting achievements for exploring new activities or attendance. Overall, this project has a great potential for becoming a full-fledged multifunctional children's development application that can make well-rounded education even more accessible.

References

Mercado, A. (2025, April 7). *Coursera statistics: The growth of e-learning in 2025*. Skillademia. https://www.skillademia.com/statistics/coursera-statistics/

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural Collaborative Filtering. Retrieved April 25, 2025, from arXiv.org website: https://arxiv.org/abs/1708.05031

Appendix A: Source Code Repositories

Backend Microservices

1. *Main Service* – GitLab Repository. Available at: https://gitlab.com/team.wombats.nani/main-service

2. *GraphQL Gateway* – GitLab Repository. Available at: https://gitlab.com/team.wombats.nani/graphql-gateway

3. *Keycloak OTP Provider* – GitLab Repository. Available at: https://gitlab.com/team.wombats.nani/keycloak-otp-provider

4. *Message Sender* – GitLab Repository. Available at: https://gitlab.com/team.wombats.nani/message-sender

5. Admin Web Page – GitLab Repository. Available at: https://gitlab.com/skolder/nani-front