:AudioWave
A Microservices-Based Audio Sharing Platform
Design and Implementation of a Full Microservices Architecture

By

Ward Zidani

Ahmad Bsese

Project Code: 24-1-D-44

Software Engineering Department

Braude - College of Engineering, Karmiel **Summer 2024**

Problem and Motivation

- **Problem**: There is a growing demand for platforms that allow users to share and listen to audio content efficiently.

- **Motivation**: The need for a scalable and flexible solution that can handle dynamic traffic while providing a seamless audio streaming experience.

System Requirements

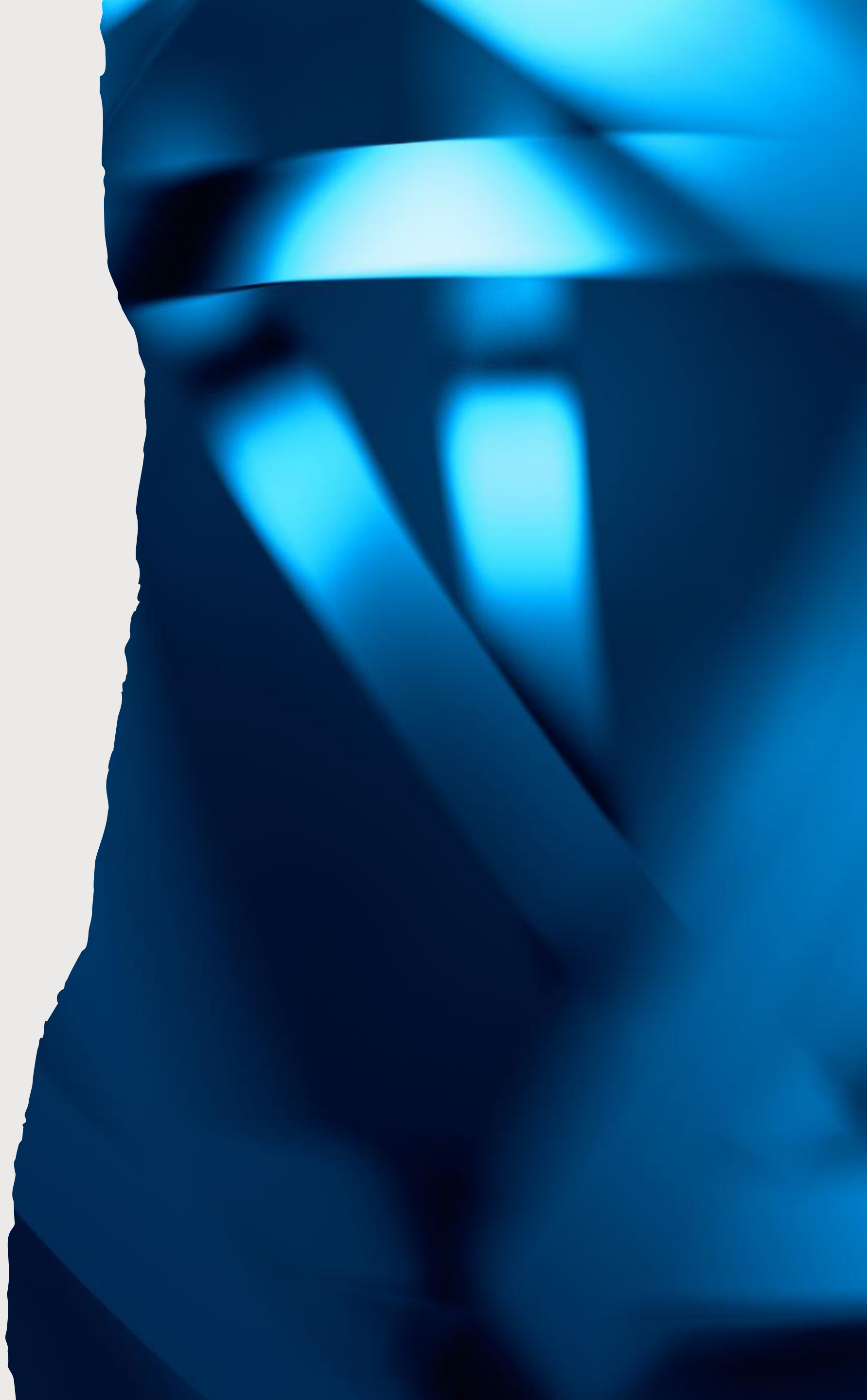
Scalability: The system should handle thousands of users simultaneously.

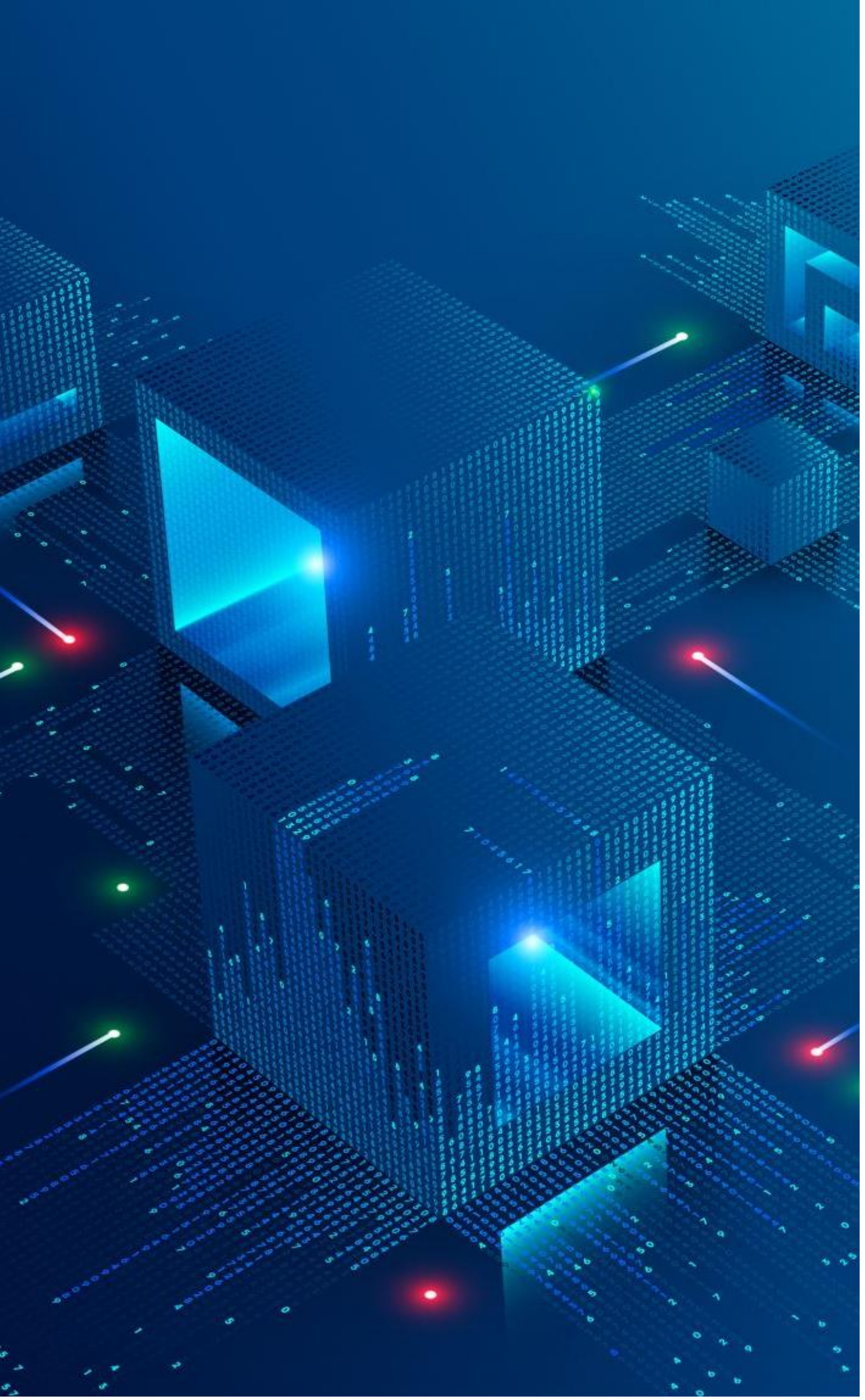
Security: User data must be encrypted and securely managed.

Efficiency: Audio uploads and playback should be fast and responsive.

Modular Design: Each service should be independent, enabling easier maintenance and scalability.

User-friendly Interface: Both content creators and consumers should find the platform easy to use.





Solution and Technologies

- **Architecture** secivresorcim gnisu tliub si metsys ehT : noitazireniatnoc rekcoD htiw erutcetihcra
- **API Gateway** eganam ot yxorp esrever a sa desu si xnigN : stseuqer gnimocni
- **Message Broker** suonorhcnysa rof desu si QMtibbaR : secivresorcim neewteb noitacinummoc
- **Databases** ,atadatem dna atad resu rof desu si revreS LQS : stsilyalp seldnah BDognoM elihw
- **Frontend**ppa elibom eht rof rettulF :
- **Encryption**BDognoM ni derots syek noitpyrcne laitnederC :

Metrics and Success Criteria

- **Scalability:** 100,000 eldnah nac metsys ehT :users concurrently.
- **Uptime**99.9% :availability due to fault tolerance.
- **Performance** elbaeciton tuohtiw maerts selfi oiduA : .syaled
- **Security** era ,slaitnederc sa hcus ,atad evitisnes llA : .detpyrcne

Challenges Faced

- **Frontend Issues**: Error when building the Flutter project in Android Studio; the solution was to open Android Studio as a system administrator.
- **Efficient Audio Streaming**: Handling large audio files without delay; the solution was to split audio into smaller chunks for streaming.
- **Fault Tolerance**: Ensuring one service failure doesn't bring down the whole system; the solution was to decouple services and use RabbitMQ.

Development Phases and Tools

- **Planning and Design**: Created the system architecture using microservices and defined key functionalities.
- **Frontend Development**: Built the mobile app using Flutter.
- **Backend Development**: Developed the microservices using .NET Core, containerized with Docker.
- **Testing**: Conducted functional and performance tests to ensure system stability.
- **Tools**: GitHub for version control, Docker for containerization, Nginx for routing, RabbitMQ for communication.

Pre-recorded Video

Link or the video

Takeaways and Improvements

- **What Worked**: The microservices architecture allowed us to scale services independently.
- **What We Would Improve**: Optimize caching strategies and better handle frontend build issues earlier in the project.
- **Key Lesson**: Early planning of architecture and communication between services is crucial for system scalability.

Conclusion and Future Development

- **Project Success**: AudioWave successfully meets its goals for scalability, performance, and user experience.
- **Future Enhancements**: Adding more features like advanced search, further optimizing performance, and potentially launching web-based clients in addition to the mobile app.

Thanks for listening