MicroFit

What is MicroFit?

MicroFit is a full-stack fitness application built using microservices architecture.

It features AI-driven personalized workout recommendations, tracks user activities securely, and leverages asynchronous messaging for scalable

and flexible operations. The backend uses Spring Boot microservices, while the frontend is developed with React.

Why I Built This Project

This project was created to demonstrate how modern microservices architecture can be applied in the fitness domain, integrating AI to offer personalized fitness guidance and scalable, maintainable software.

Problem Statement / Need

Most traditional fitness apps lack modular architecture and real-time AI-powered recommendations. There is a growing need

for an intelligent, scalable system that provides personal fitness insights based on detailed user activity data.

Key Features

- Independent microservices for user management, activity tracking, and AI recommendations

- AI-powered exercise analysis and personalized guidance

- Secure authentication with Keycloak using OAuth2 and JWT

- Asynchronous communication with RabbitMQ message queues

- Separate databases for microservices to ensure decoupled data management

- Responsive React frontend for smooth user experience

- Real-time feedback and fitness insights

Technologies Used (Tech Stack)

- Backend: Spring Boot Microservices
- Frontend: React, JavaScript
- Messaging: RabbitMQ
- Security: Keycloak, OAuth2, JWT
- Databases: MySQL, MongoDB
- AI: Google Gemini API
- Service Discovery and Routing: Eureka API Gateway
- Configuration Management: Spring Cloud Config Server

Learning Outcomes

- Building scalable microservices-based applications
- Integrating AI services with asynchronous messaging
- Implementing secure OAuth2 authentication in microservices
- Developing a responsive frontend connected to microservices
- Managing service discovery and configurations in distributed systems

Challenges Overcome

- Coordinating synchronous and asynchronous communication
- Ensuring secure, token-based user authentication
- Integrating third-party AI APIs effectively
- Managing distributed databases and microservices orchestration
- Maintaining seamless frontend-backend communication

Limitations

- Basic frontend UI, with room for aesthetic enhancements
- Limited fitness metrics (missing pace, distance, heart rate sensors)

- Dependent on cloud infrastructure for scalability

- Requires internet connectivity for AI-powered features

Future Enhancements

- Support for additional fitness metrics like pace and heart rate

- Enhanced UI/UX design and interactive data visualizations

- Extended AI model capabilities for comprehensive fitness insights

- Cloud-native deployment with Kubernetes and Docker

- Integration with wearable fitness devices for live tracking