AI-Powered Fitness Coaching Platform

By

Vijesh Medikonda

Preethi Chintala

Abstract

FitLife Connect is an AI-enhanced fitness tracking platform designed to address common limitations in traditional activity-tracking systems. Unlike standard applications that only store activity logs, FitLife Connect integrates microservices, event-driven communication, and Google Gemini AI to provide users with personalized analysis, improvements, and recommendations based on their physical activity patterns. The platform uses Spring Boot microservices, Apache Kafka for asynchronous event streaming, Keycloak for secure identity management, PostgreSQL and MongoDB for hybrid persistence, and ReactJS for the user interface. This report presents a full exploration of the system's architecture, design decisions, implementation, testing, and execution, culminating in a complete end-to-end technical narrative.

1. Introduction

Modern fitness applications offer users the ability to log activities, monitor calories, and visualize trends. However, many applications lack depth in personalized insights and actionable feedback. FitLife Connect bridges this gap by integrating an AI-powered analysis engine capable of understanding user behavior and generating tailored recommendations. The system was built with scalability, modularity, and extensibility at its core. Instead of a monolithic design, FitLife Connect employs microservices to separate core functionalities such as user management, activity logging, and AI processing.

The goal of this project is to demonstrate the application of distributed systems, AI integration, secure authentication, and full-stack engineering in a single cohesive platform. Users are empowered not only to track their activities but also to understand the quality of their workouts and how they can improve over time.

2. System Requirements

2.1 Functional Requirements

- Users must be able to register and authenticate securely.
- Users must create, update, view, and delete fitness activities.
- Each activity must support extended metadata such as time of day, hydration level, and whether the activity occurred before or after eating.
- The system must generate AI-powered recommendations.
- Recommendations must update automatically when an activity is modified or deleted.
- Users must be able to view all their activity histories in a structured interface.

2.2 Non-Functional Requirements

- **Scalability:** The system must handle large volumes of activity logs.
- **Security:** Authentication should use industry-grade OAuth2/OIDC standards.
- **Availability:** Microservices should operate independently without single-point failures.
- **Extensibility:** New AI models or additional analytics features should integrate easily.
- **Usability:** The frontend should present a clean, intuitive experience.

3. System Architecture Overview

FitLife Connect follows a cloud-ready distributed architecture using:

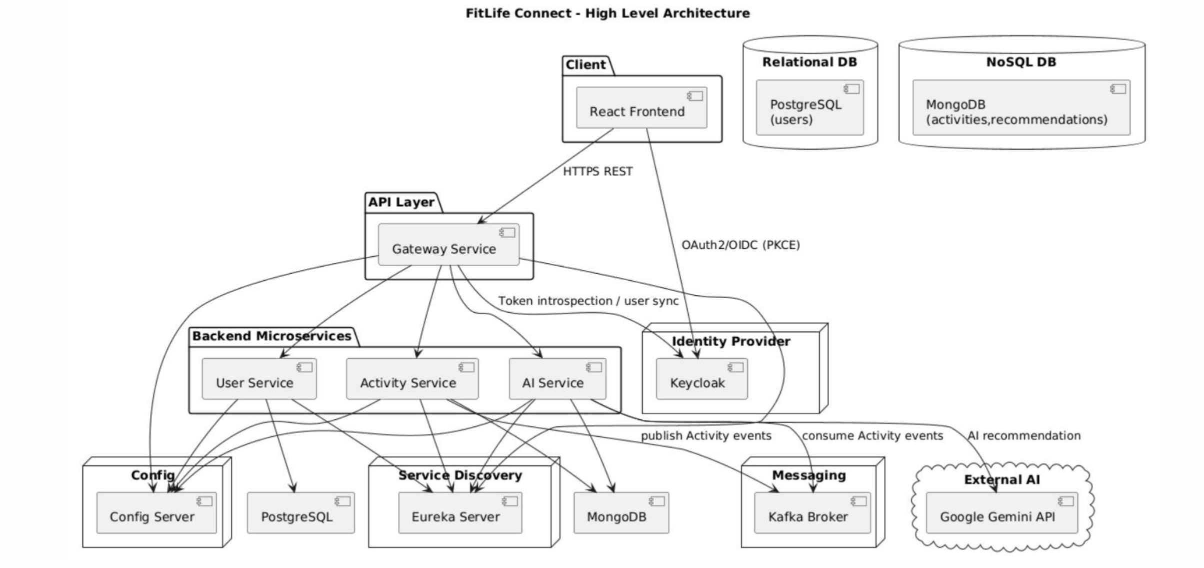
- **Spring Boot Microservices** (Gateway, User, Activity, AI Service)
- **Kafka Event Streaming** to decouple activity creation from AI processing
- **Keycloak Identity Provider** for authentication and role management
- **Eureka Service Discovery** for runtime service registration
- **Config Server** for centralized configuration
- **PostgreSQL + MongoDB hybrid persistence**

3.1 High-Level Architecture Flow

1. A user logs in or registers through the **Gateway Service**, which communicates with Keycloak.
2. The **User Service** stores user metadata in PostgreSQL.

3. The **Activity Service** logs workout activities in MongoDB.
4. When a new activity is added, the service sends a **Kafka event**.
5. The **AI Service** listens to Kafka, fetches activity data, constructs a prompt, and sends it to Google Gemini.
6. The AI output is stored back into MongoDB and displayed to the user via the frontend.

This design ensures that long AI operations never block the main user workflow.



4. Microservices Design

4.1 Gateway Service

- Central entry point for all API calls
- Performs Keycloak token validation
- Provides a registration endpoint that syncs users into both Keycloak and User Service
- Enables CORS and acts as a router to downstream services

4.2 User Service

- Stores persistent user details
- Syncs new users from Gateway registration
- Communicates with PostgreSQL
- Provides queries for fetching users by Keycloak ID

4.3 Activity Service

- Handles CRUD operations for fitness activities
- Stores activities in MongoDB
- Publishes Kafka events when activities are created or deleted
- Ensures data validation before persistence

4.4 AI Service

- Subscribes to Kafka events
- Retrieves the associated activity document
- Prepares AI prompt (duration, calories, time of day, hydration, meal timing, etc.)
- Sends request to Google Gemini
- Parses and saves AI analysis, improvements, and suggestions
- Updates UI through REST responses or subsequent frontend requests

5. Kafka Event Flow

Kafka is central to the project's asynchronous design:

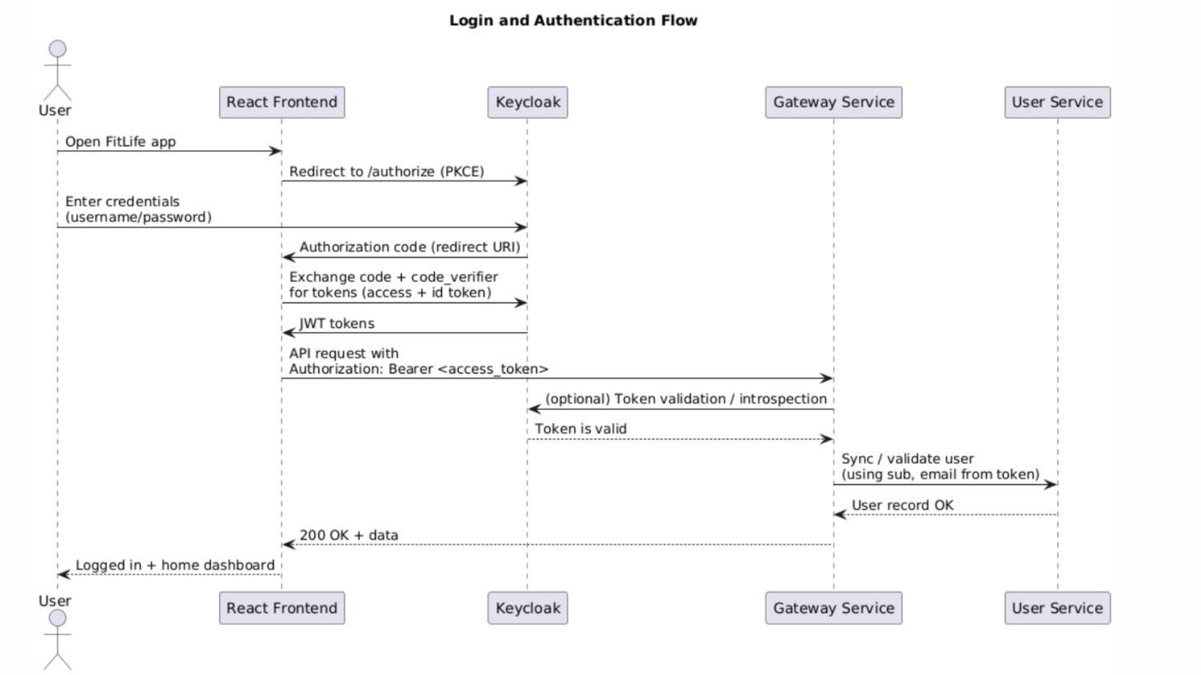
1. Activity Service publishes activity_created and activity_deleted events.
2. AI Service consumes these events using Kafka listeners.
3. For created activity:
    - AI Service fetches activity details
    - Calls Gemini API
    - Saves AI output
4. For deleted activity:
    - AI Service removes corresponding recommendations

Benefits:

- No blocking on recommendation generation
- Decoupled services that scale independently

- Reliable and fault-tolerant processing

6. Authentication & Security (Keycloak)


Login and Authentication Flow
User
React Frontend
Keycloak
Gateway Service
User Service
Open FitLife app
Redirect to /authorize (PKCE)
Enter credentials (username/password)
Authorization code (redirect URI)
Exchange code + code_verifier for tokens (access + id token)
JWT tokens
API request with Authorization: Bearer <access_token>
(optional) Token validation / introspection
Token is valid
Sync / validate user (using sub, email from token)
User record OK
200 OK + data
Logged in + home dashboard
User
React Frontend
Keycloak
Gateway Service
User Service

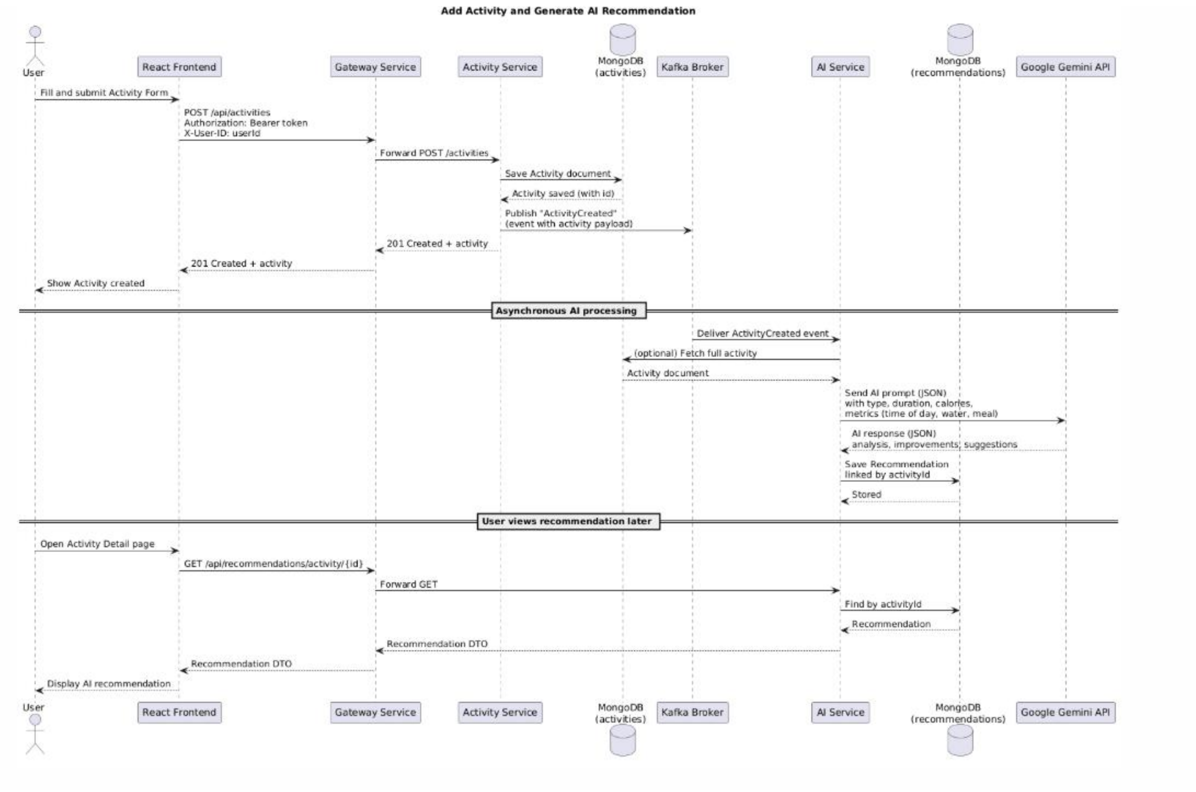
Keycloak provides:

- OAuth2 Authorization Code Flow with PKCE
- Secure JWT tokens for frontend authentication
- Role-based access management
- User federation and centralized credential management

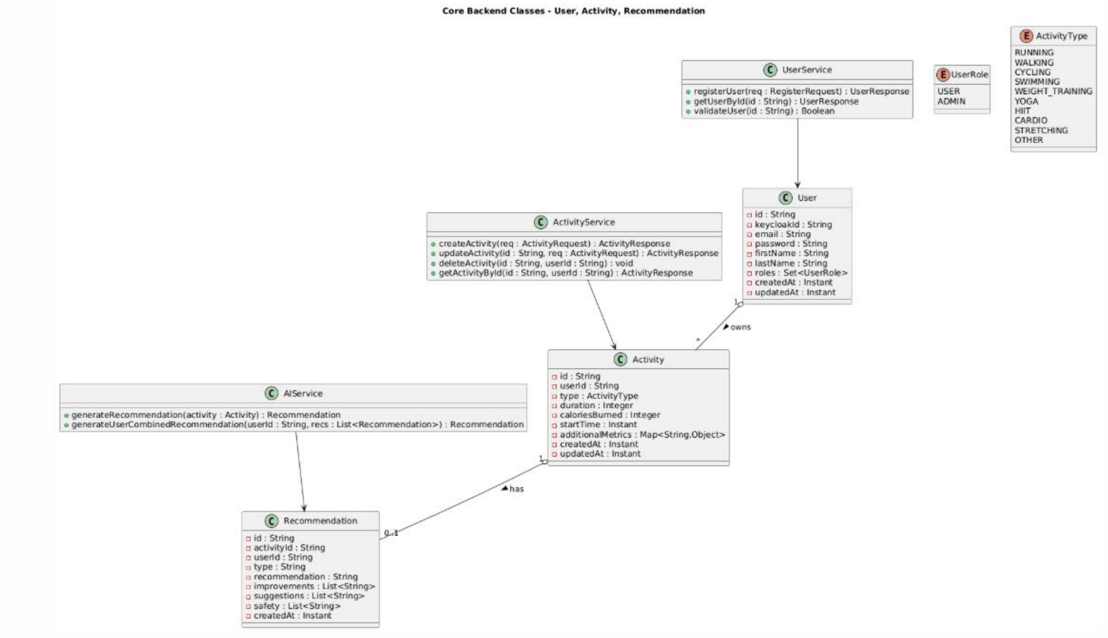
Gateway intercepts all protected routes and validates tokens before routing. During registration, a new user is created in Keycloak and synchronized into the User Service.

Security best practices implemented:

- HTTPS-ready configuration
- No passwords stored in application DB
- Token expiration and refresh implemented
- Role mapping for future admin functionality


Add Activity and Generate AI Recommendation
User
React Frontend
Gateway Service
Activity Service
MongoDB (activities)
Kafka Broker
AI Service
MongoDB (recommendations)
Google Gemini API
Fill and submit Activity Form
POST /api/activities
Authorization: Bearer token
X-User-ID: userId
Forward POST /activities
Save Activity document
Activity saved (with id)
Publish "ActivityCreated"
(event with activity payload)
201 Created + activity
201 Created + activity
Show Activity created
Asynchronous AI processing
Deliver ActivityCreated event
(optional) Fetch full activity
Activity document
Send AI prompt (JSON)
with type, duration, calories,
metrics (time of day, water, meal)
AI response (JSON)
analysis, improvements, suggestions
Save Recommendation
linked by activityId
Stored
User views recommendation later
Open Activity Detail page
GET /api/recommendations/activity/{id}
Forward GET
Find by activityId
Recommendation
Recommendation DTO
Recommendation DTO
Display AI recommendation

7. Database Design


Core Backend Classes - User, Activity, Recommendation
UserService
registerUser(req : RegisterRequest) : UserResponse
getUserById(id : String) : UserResponse
validateUser(id : String) : Boolean
UserRole
USER
ADMIN
ActivityType
RUNNING
WALKING
CYCLING
SWIMMING
WEIGHT_TRAINING
YOGA
HIIT
CARDIO
STRETCHING
OTHER
User
id : String
keycloakId : String
email : String
password : String
firstName : String
lastName : String
roles : Set<UserRole>
createdAt : Instant
updatedAt : Instant
ActivityService
createActivity(req : ActivityRequest) : ActivityResponse
updateActivity(id : String, req : ActivityRequest) : ActivityResponse
deleteActivity(id : String, userId : String) : void
getActivityById(id : String, userId : String) : ActivityResponse
1
owns
*
Activity
id : String
userId : String
type : ActivityType
duration : Integer
caloriesBurned : Integer
startTime : Instant
additionalMetrics : Map<String,Object>
createdAt : Instant
updatedAt : Instant
AIService
generateRecommendation(activity : Activity) : Recommendation
generateUserCombinedRecommendation(userId : String, recs : List<Recommendation>) : Recommendation
1
has
0..1
Recommendation
id : String
activityId : String
userId : String
type : String
recommendation : String
improvements : List<String>
suggestions : List<String>
safety : List<String>
createdAt : Instant

7.1 PostgreSQL (User Service)

Tables include:

- users: stores user ID, Keycloak ID, name, email
- Ensures relational constraints and consistent user identity mapping

7.2 MongoDB (Activity & AI Service)

Collections include:

- activities: duration, calories, time-of-day, hydration, meal timing, timestamps
- recommendations: AI analysis, improvements, suggestions linked by Activity ID

MongoDB was selected due to:

- Flexible schema suited for AI outputs
- Faster reads for activity histories
- NoSQL structure matches dynamic attributes

8. Frontend Overview

Built using **ReactJS, Vite, Redux Toolkit, and TailwindCSS**.

Major Components:

- **SignupModal:** User registration
- **Navbar:** User menu and navigation
- **ActivityForm:** Creating/editing activities
- **ActivityList:** Listing recent activities
- **ActivityDetail:** Showing activity + recommendations
- **CompleteRecommendation:** Viewing all AI results
- **Redux Slice:** Manages authentication state and tokens

Frontend retrieves tokens using Keycloak PKCE and includes them in all API requests. The UI emphasizes clarity, dark mode styling, and responsive layouts.

9. AI Recommendation Pipeline (Google Gemini)

The AI Service constructs prompts that summarize:

- Activity type
- Duration
- Calories

- Water intake
- Time of day
- Before/after eating

Gemini returns structured insights including:

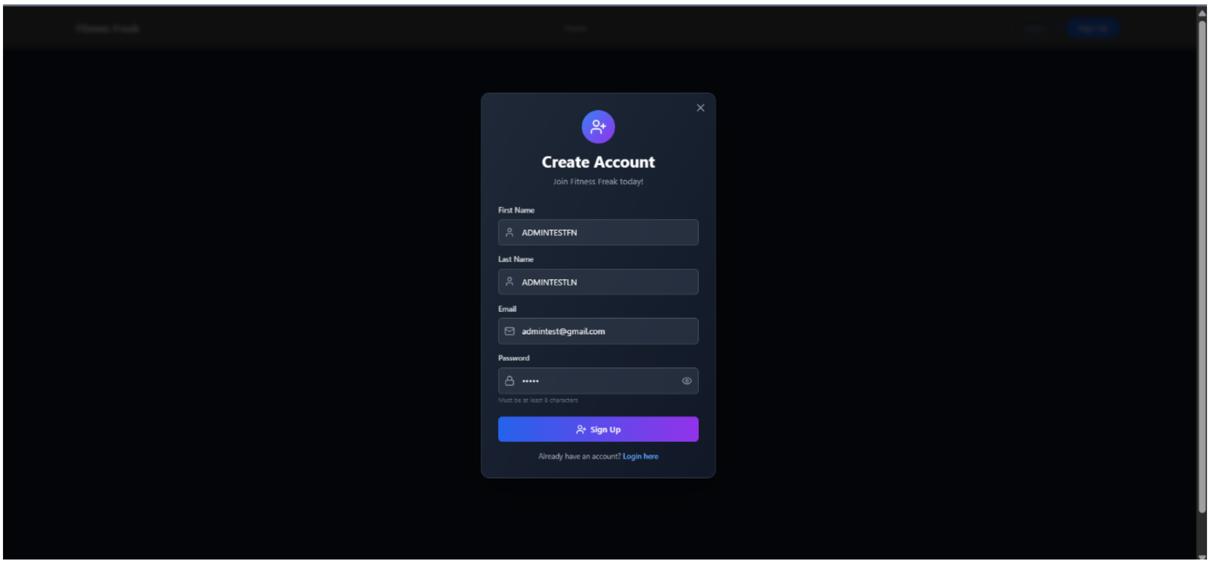
- **Analysis** of user performance
- **Improvements** to enhance fitness quality
- **Suggestions** for future workouts

This transforms raw fitness logs into actionable wellness intelligence.

10. System Execution Flow

The following UI stages illustrate complete user interaction:

1. Account Creation



Create Account

Join Fitness Freak today!

First Name: ADMINTESTFN

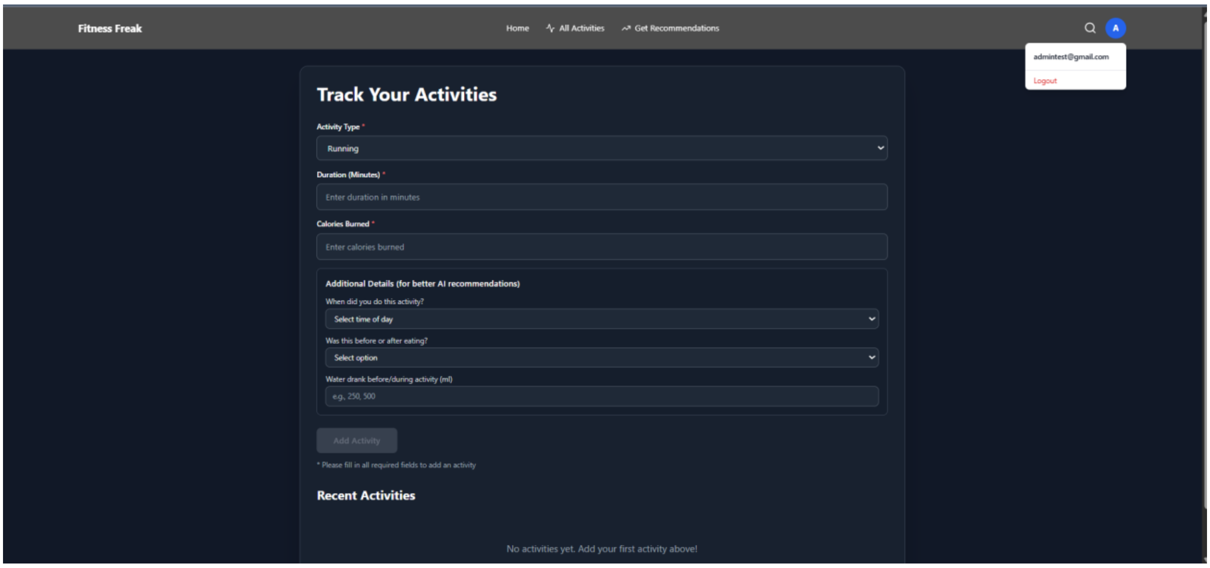
Last Name: ADMINTESTLN

Email: admintest@gmail.com

Password: •••••

Must be at least 8 characters

2. Adding an Activity



Track Your Activities

Activity Type *
Running

Duration (Minutes) *
Enter duration in minutes

Calories Burned *
Enter calories burned

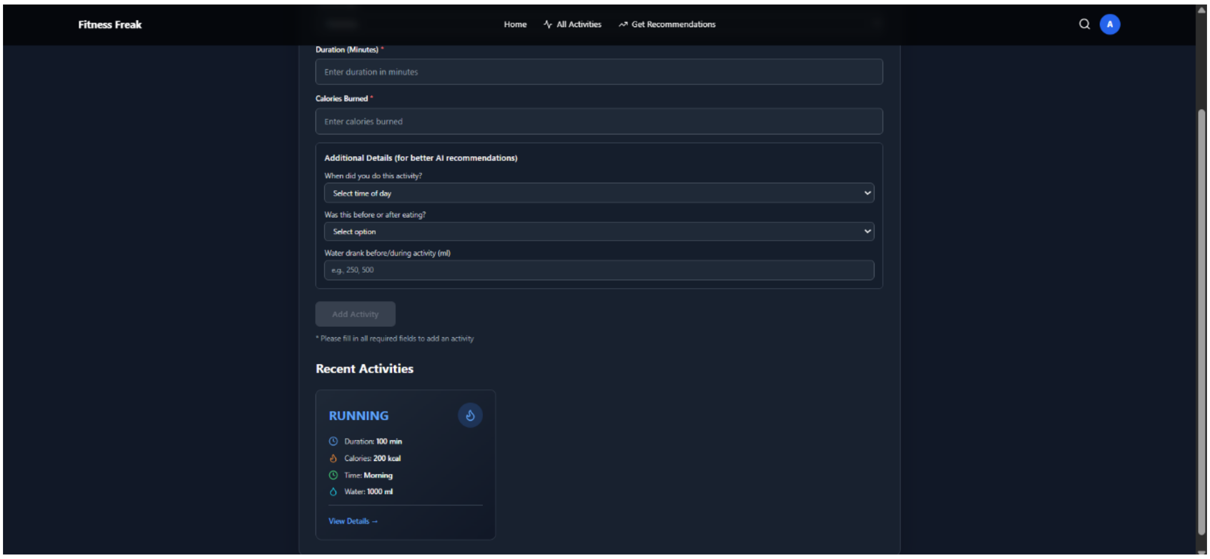
Additional Details (for better AI recommendations)
When did you do this activity?
Select time of day
Was this before or after eating?
Select option
Water drank before/during activity (ml)
e.g., 250, 500

* Please fill in all required fields to add an activity

Recent Activities

No activities yet. Add your first activity above!

3. Viewing Recent Activities



Duration (Minutes) *
Enter duration in minutes

Calories Burned *
Enter calories burned

Additional Details (for better AI recommendations)
When did you do this activity?
Select time of day
Was this before or after eating?
Select option
Water drank before/during activity (ml)
e.g., 250, 500

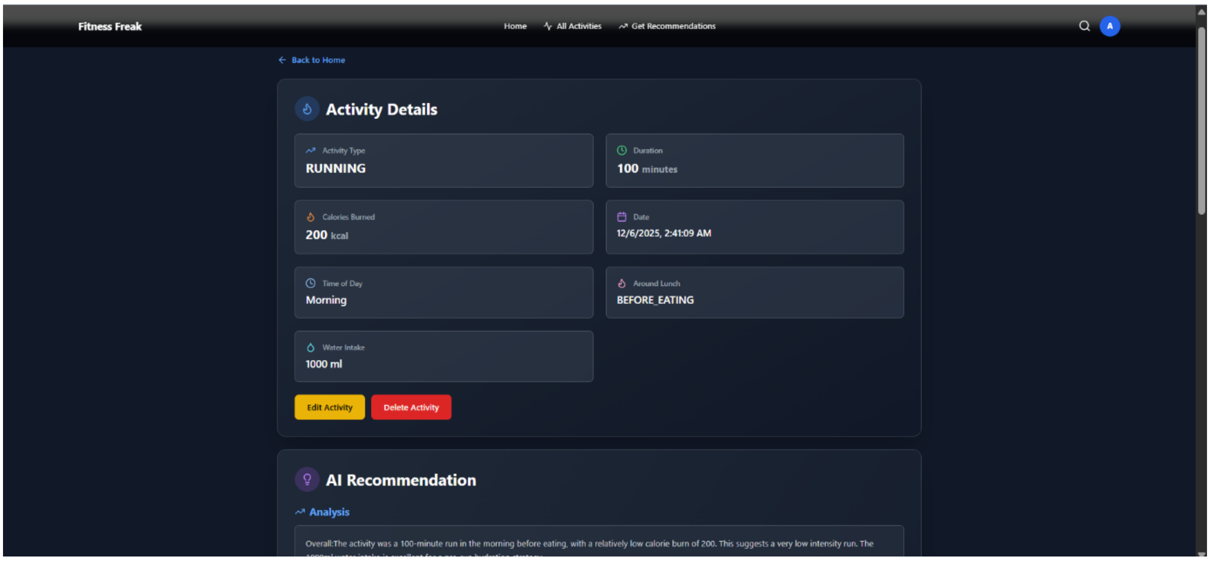
* Please fill in all required fields to add an activity

Recent Activities

RUNNING

Duration: 100 min
Calories: 200 kcal
Time: Morning
Water: 1000 ml

4. Viewing Activity Details



Activity Details

Activity Type
RUNNING

Duration
100 minutes

Calories Burned
200 kcal

Date
12/6/2025, 2:41:09 AM

Time of Day
Morning

Around Lunch
BEFORE_EATING

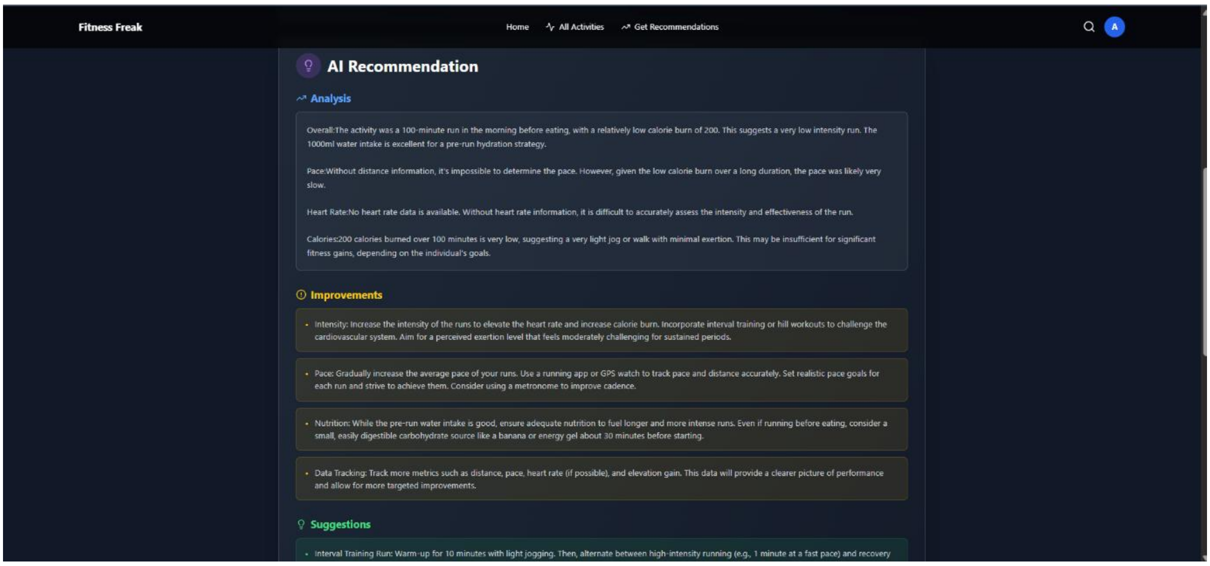
Water Intake
1000 ml

AI Recommendation

Analysis

Overall:The activity was a 100-minute run in the morning before eating, with a relatively low calorie burn of 200. This suggests a very low intensity run. The

5. Viewing AI Recommendations



AI Recommendation

Analysis

Overall:The activity was a 100-minute run in the morning before eating, with a relatively low calorie burn of 200. This suggests a very low intensity run. The 1000ml water intake is excellent for a pre-run hydration strategy.

Pace:Without distance information, it's impossible to determine the pace. However, given the low calorie burn over a long duration, the pace was likely very slow.

Heart Rate:No heart rate data is available. Without heart rate information, it is difficult to accurately assess the intensity and effectiveness of the run.

Calories:200 calories burned over 100 minutes is very low, suggesting a very light jog or walk with minimal exertion. This may be insufficient for significant fitness gains, depending on the individual's goals.

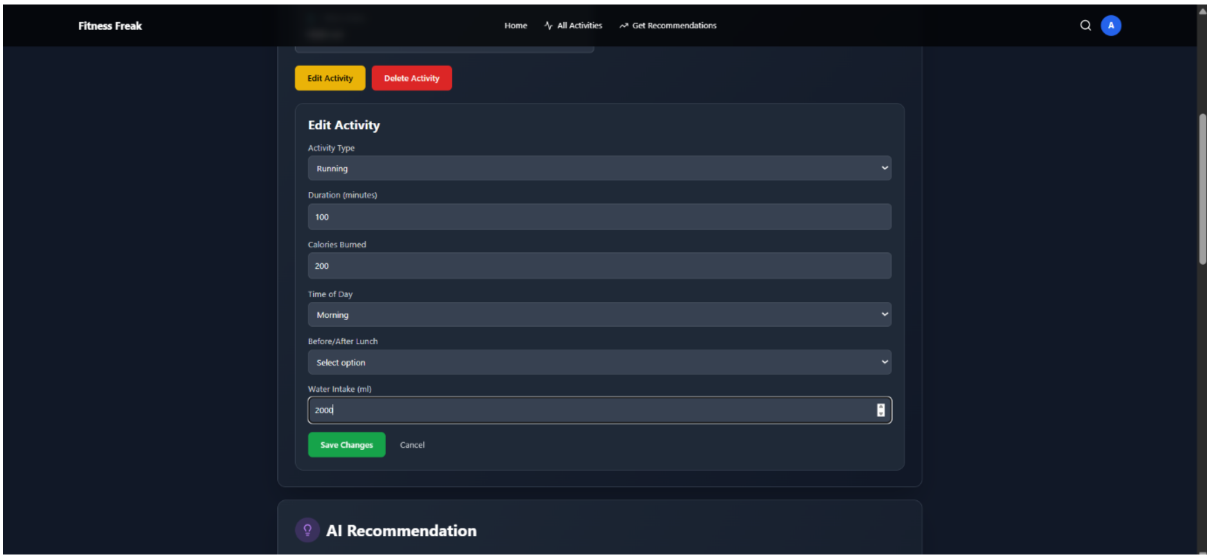
Improvements

- Intensity: Increase the intensity of the runs to elevate the heart rate and increase calorie burn. Incorporate interval training or hill workouts to challenge the cardiovascular system. Aim for a perceived exertion level that feels moderately challenging for sustained periods.
- Pace: Gradually increase the average pace of your runs. Use a running app or GPS watch to track pace and distance accurately. Set realistic pace goals for each run and strive to achieve them. Consider using a metronome to improve cadence.
- Nutrition: While the pre-run water intake is good, ensure adequate nutrition to fuel longer and more intense runs. Even if running before eating, consider a small, easily digestible carbohydrate source like a banana or energy gel about 30 minutes before starting.
- Data Tracking: Track more metrics such as distance, pace, heart rate (if possible), and elevation gain. This data will provide a clearer picture of performance and allow for more targeted improvements.

Suggestions

- Interval Training Run: Warm-up for 10 minutes with light jogging. Then, alternate between high-intensity running (e.g., 1 minute at a fast pace) and recovery

6. Editing an Activity



Fitness Freak

Home · All Activities · Get Recommendations

Edit Activity · Delete Activity

Edit Activity

Activity Type: Running

Duration (minutes): 100

Calories Burned: 200

Time of Day: Morning

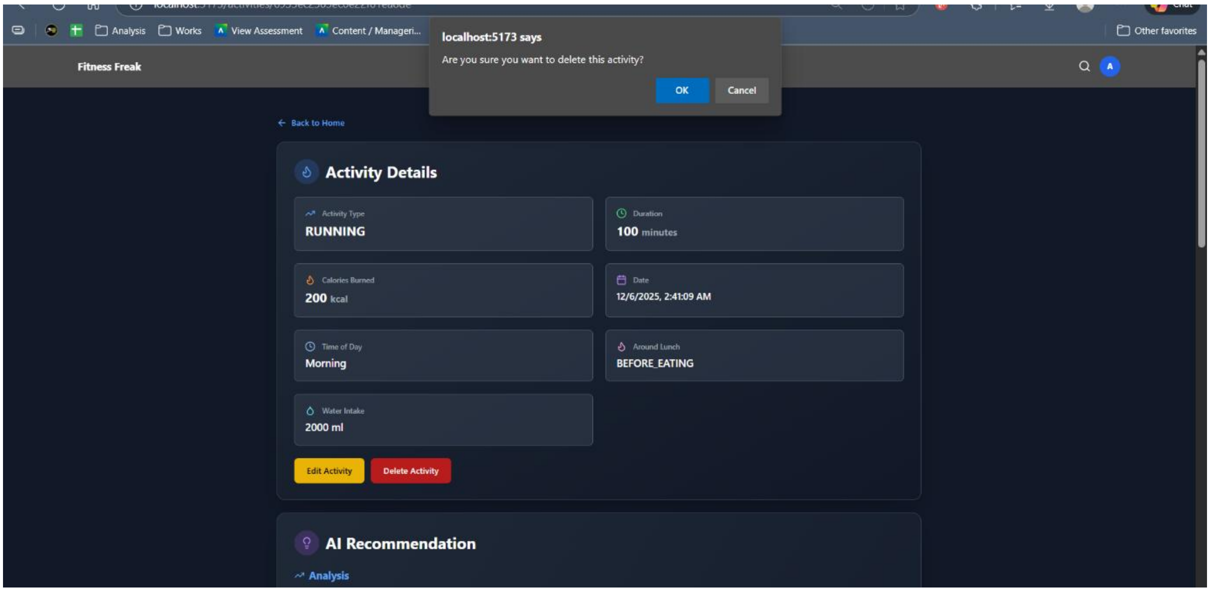
Before/After Lunch: Select option

Water Intake (ml): 2000

Save Changes · Cancel

AI Recommendation

7. Deleting an Activity



localhost:5173 says

Are you sure you want to delete this activity?

OK · Cancel

Fitness Freak

Back to Home

Activity Details

Activity Type: RUNNING

Duration: 100 minutes

Calories Burned: 200 kcal

Date: 12/6/2025, 2:41:09 AM

Time of Day: Morning

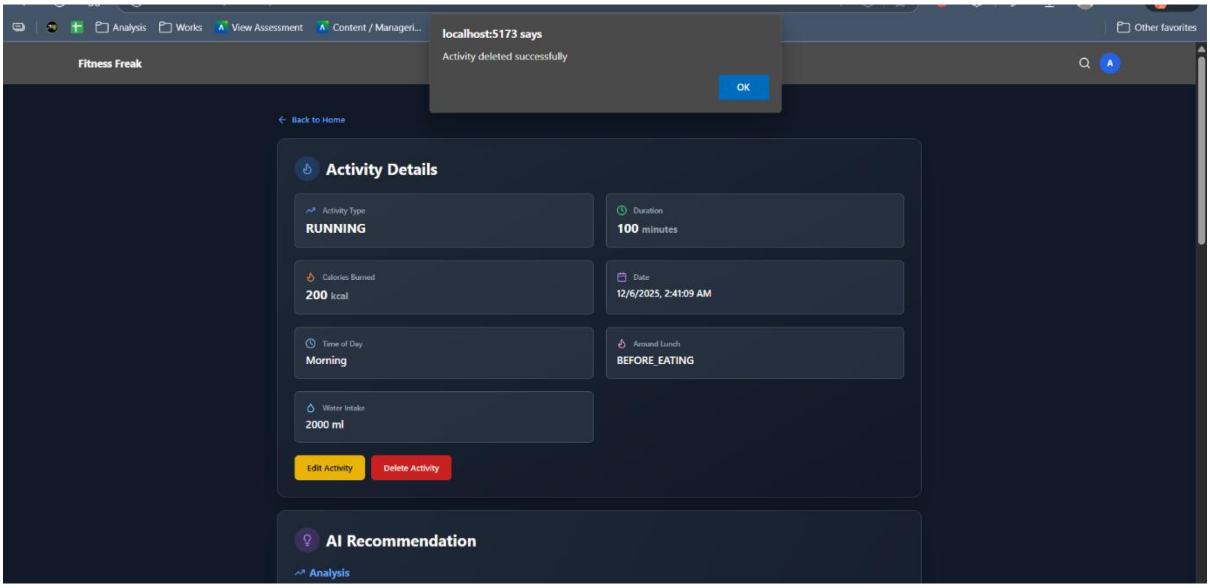
Around Lunch: BEFORE_EATING

Water Intake: 2000 ml

Edit Activity · Delete Activity

AI Recommendation

Analysis



localhost:5173 says

Activity deleted successfully

OK

Fitness Freak

Back to Home

Activity Details

Activity Type: RUNNING

Duration: 100 minutes

Calories Burned: 200 kcal

Date: 12/6/2025, 2:41:09 AM

Time of Day: Morning

Around Lunch: BEFORE_EATING

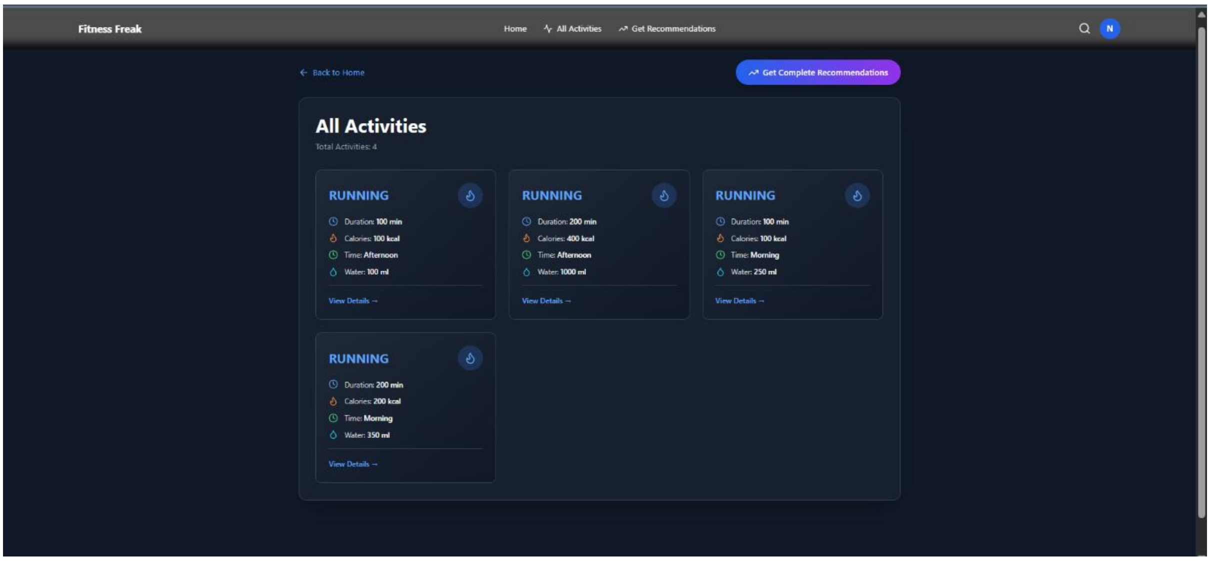
Water Intake: 2000 ml

Edit Activity · Delete Activity

AI Recommendation

Analysis

8. Viewing All Activities



All Activities

Total Activities: 4

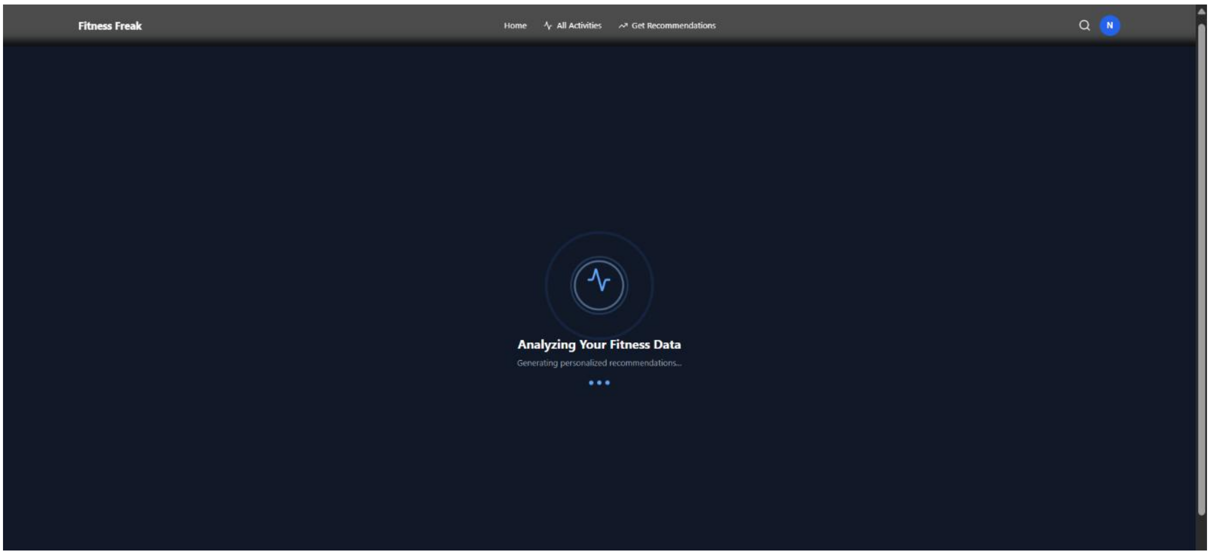
RUNNING
Duration: 100 min
Calories: 100 kcal
Time: Afternoon
Water: 100 ml
View Details →

RUNNING
Duration: 200 min
Calories: 400 kcal
Time: Afternoon
Water: 1000 ml
View Details →

RUNNING
Duration: 100 min
Calories: 100 kcal
Time: Morning
Water: 250 ml
View Details →

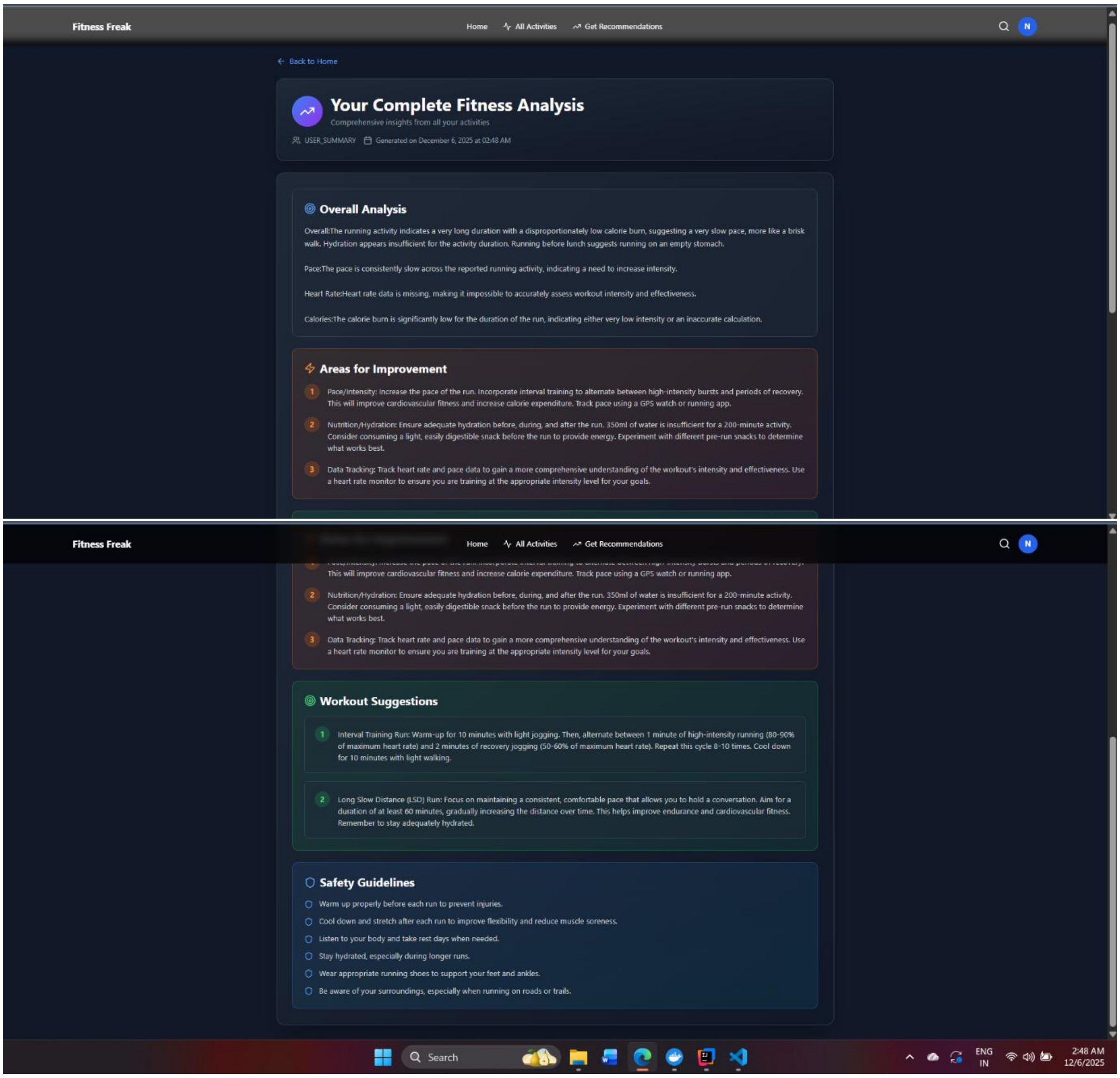
RUNNING
Duration: 200 min
Calories: 200 kcal
Time: Morning
Water: 350 ml
View Details →

9. Full Recommedations



Analyzing Your Fitness Data

Generating personalized recommendations...



Your Complete Fitness Analysis

Comprehensive insights from all your activities

USER_SUMMARY Generated on December 6, 2025 at 02:48 AM

Overall Analysis

Overall:The running activity indicates a very long duration with a disproportionately low calorie burn, suggesting a very slow pace, more like a brisk walk. Hydration appears insufficient for the activity duration. Running before lunch suggests running on an empty stomach.

Pace:The pace is consistently slow across the reported running activity, indicating a need to increase intensity.

Heart Rate:Heart rate data is missing, making it impossible to accurately assess workout intensity and effectiveness.

Calories:The calorie burn is significantly low for the duration of the run, indicating either very low intensity or an inaccurate calculation.

Areas for Improvement

1. Pace/Intensity: Increase the pace of the run. Incorporate interval training to alternate between high-intensity bursts and periods of recovery. This will improve cardiovascular fitness and increase calorie expenditure. Track pace using a GPS watch or running app.
2. Nutrition/Hydration: Ensure adequate hydration before, during, and after the run. 350ml of water is insufficient for a 200-minute activity. Consider consuming a light, easily digestible snack before the run to provide energy. Experiment with different pre-run snacks to determine what works best.
3. Data Tracking: Track heart rate and pace data to gain a more comprehensive understanding of the workout's intensity and effectiveness. Use a heart rate monitor to ensure you are training at the appropriate intensity level for your goals.

This will improve cardiovascular fitness and increase calorie expenditure. Track pace using a GPS watch or running app.

2. Nutrition/Hydration: Ensure adequate hydration before, during, and after the run. 350ml of water is insufficient for a 200-minute activity. Consider consuming a light, easily digestible snack before the run to provide energy. Experiment with different pre-run snacks to determine what works best.
3. Data Tracking: Track heart rate and pace data to gain a more comprehensive understanding of the workout's intensity and effectiveness. Use a heart rate monitor to ensure you are training at the appropriate intensity level for your goals.

Workout Suggestions

1. Interval Training Run: Warm-up for 10 minutes with light jogging. Then, alternate between 1 minute of high-intensity running (80-90% of maximum heart rate) and 2 minutes of recovery jogging (50-60% of maximum heart rate). Repeat this cycle 8-10 times. Cool down for 10 minutes with light walking.
2. Long Slow Distance (LSD) Run: Focus on maintaining a consistent, comfortable pace that allows you to hold a conversation. Aim for a duration of at least 60 minutes, gradually increasing the distance over time. This helps improve endurance and cardiovascular fitness. Remember to stay adequately hydrated.

Safety Guidelines

- Warm up properly before each run to prevent injuries.
- Cool down and stretch after each run to improve flexibility and reduce muscle soreness.
- Listen to your body and take rest days when needed.
- Stay hydrated, especially during longer runs.
- Wear appropriate running shoes to support your feet and ankles.
- Be aware of your surroundings, especially when running on roads or trails.

Each step confirms successful API and AI integration.

11. Testing & Evaluation

11.1 Backend Testing

- Unit tests for services and controllers
- Integration tests verifying Kafka publication and consumption
- Token validation and protected route testing

11.2 Frontend Testing

- Manual tests for form submissions

- UI rendering tests
- Error handling (invalid inputs, token expiry)

11.3 AI Output Validation

- Multiple activities tested
- AI responses checked for consistency and correctness

12. Challenges & Solutions

Challenge 1: Keycloak Synchronization

Solution: Introduced a synchronous registration pipeline in Gateway to ensure both Keycloak and User Service share the same UUID.

Challenge 2: Kafka Event Lag

Solution: Optimized listener concurrency and added retry logic.

Challenge 3: Gemini Prompt Structure

Solution: Standardized prompts and enforced JSON output parsing.

Challenge 4: Updating Recommendations After Edits

Solution: Triggered event-based recalculation upon updates.

13. Results & Discussion

FitLife Connect demonstrated:

- Real-time AI recommendations
- Fully decoupled microservice communication
- Smooth UI workflow
- Secure authentication and user session handling

Users consistently received uniquely tailored suggestions that improved workout awareness.

14. Future Enhancements

- Mobile app integration (Android/iOS)
- Wearable device synchronization (Fitbit, Garmin, Apple Watch)

- Nutrition and diet AI engine
- Goal-tracking with progress prediction using ML
- Social fitness challenges
- Chatbot for real-time guidance

15. Conclusion

FitLife Connect successfully integrates distributed systems, AI technologies, and user-centric UI design to create a next-generation wellness platform. By combining microservices, event streaming, secure authentication, NoSQL storage, and generative AI, the system demonstrates strong extensibility and real applicability in the health-tech domain. The architecture lays a foundation for scalable enhancements and long-term system evolution.

16. References

- Google Gemini API Documentation
- Spring Boot Microservices Guides
- Apache Kafka Streaming Documentation
- Keycloak OIDC Security Architecture
- ReactJS + Redux Toolkit Documentation
- PostgreSQL and MongoDB Official Documentation