



Software Engineering B.Sc. Final Project

Authors:

Maayan Babayof - 207000415

Lotem Yakovian - 312328537

Nikolai Malanichev - 342429388

Supervisors:

Dr. Marcelo Sihman,

Dr. Dvir Ross

31/08/2025

Approved by: Marcelo Sihman Date: 31/08/2025 ז' אלול התשפ"ה

Approved by: Dr. Dvir Ross Date: 31/08/2025 ז' אלול התשפ"ה

**A thesis submitted in partial fulfilment of the requirements for the
degree of BSc. in Software Engineering**

Abstract

Running performance and workout motivation are strongly influenced by music, yet fixed playlists or random selections often fail to match the runner's physiological state. This can result in decreased efficiency, reduced enjoyment, and even increased risk of overexertion.

In this project, we present OnBeat, a personalized fitness system that adapts music playback in real time according to the runner's heart rate. By continuously monitoring the user's BPM, the system dynamically adjusts tempo and track selection to help maintain the desired training zone. This adaptive approach enhances performance, supports safe training, and promotes a more engaging running experience.

Our solution is designed for a wide range of users, from beginner runners to competitive athletes and rehabilitation patients. Trainers are also able to monitor progress and review session outcomes. Beyond real-time adaptation, OnBeat provides post-run analytics that highlight trends and measure the effectiveness of music during training.

By combining wearable technology with adaptive feedback mechanisms, we deliver a user-centered approach that demonstrates how personalized music can improve both the efficiency and enjoyment of running.

Acknowledgements

We would like to thank Dr. Marcelo Sihman and Dr. Dvir Ross for their guidance, support, and encouragement throughout the project.

Finally, we would like to thank the Head of the Department, Prof. Avivit Levy, for her support and direction during the course of our studies.

Table of Contents

1 INTRODUCTION.....	6
1.1 Problem Description	6
1.2 Motivation	6
1.3 Goals.....	7
1.4 Overview of the Approach.....	7
1.5 Usage scenarios	7
1.5.1 Runner Scenario: Personalized Music-Guided Run.....	7
1.6 Audience.....	9
1.7 Glossary	9
1.7.1 Generic terminology	9
1.7.2 OnBeat specific terminology	10
2 LITERATURE SURVEY	11
2.1 Problem Survey	11
2.2 Solution Survey	12
2.3 Discussions and Conclusions.....	13
3 TECHNOLOGICAL SURVEY	15
3.1 SQL vs NO SQL	15
3.2 Algorithm: LightGBM.....	15
3.3 Wearable Technology: Colmi R06 Smart Ring	16
3.4 Python.....	16
3.5 React native	16
4 REQUIREMENTS AND SPECIFICATION	17
4.1 OnBeat functional requirements	17
4.2 OnBeat non-Functional requirements.....	18
4.3 Specification - The Scope of the OnBeat Project	18
5 THE ARCHITECTURE.....	19
5.1 OnBeat Data	19
5.1.1 The internal and external representations	20
5.1.2 Initial start-up data	21
5.2 The OnBeat approach	22
5.2.1 Data Collection & Preprocessing.....	22
5.2.2 Song Effectiveness & Scoring	22
5.2.3 Machine Learning Model	23
5.2.4 Song Selection Process.....	23
5.2.5 Real-Time Adaptation.....	24
5.2.6 Workout Phases	24
5.2.7 Continuous Learning.....	24
5.3 OnBeat Processes	25
5.4 Roles	26
5.5 The life-cycle view of a system	27
6 SYSTEM DESIGN	29
6.1 Data components	29
6.2 Process components.....	30
6.3 Communication components (user-computer interaction)	31
6.4 Interactions	31
6.4.1 Activity diagram	31
6.4.2 Design sequence diagram	33
6.5 System Architecture	35
7 IMPLEMENTATION.....	37
7.1 Algorithm Design	37
7.2 Interfaces	41
7.2.1 Human-computer interfaces (GUI).....	41

7.2.2 The OnBeat-system configuration.....	45
7.3 Development environment.....	46
7.4 Programming languages	47
7.5 Limitations of the system.....	47
7.6 Installation and configuration issues	48
7.7 Information Security.....	48
7.8 Risk Management.....	49
7.9 Exceptions Management.....	50
7.10 Versions Control & Code	51
7.11 Project Management.....	51
8 SYSTEM VALIDATION	53
9 SUMMARY, EVALUATION, CONCLUSIONS AND FUTURE WORK.....	63
9.1 Summary.....	63
9.2 Evaluation and conclusions	63
9.3 Future work.....	63
10 REFERENCES	65

Table of Figures

Figure 1: Workout screen with a run	8
Figure 2: Admin Dashboard with Figure's 1 stats.....	9
Figure 3: BMI formula	10
Figure 4: System architecture	19
Figure 5 - BPM age table.....	22
Figure 6: Continuous Improvement.....	25
Figure 7: Life-cycle diagram	27
Figure 8: Data Model Diagram.....	29
Figure 9: Activity diagram	32
Figure 10: Sequence diagram	33
Figure 11: Use-case diagram	34
Figure 12: System architecture	35
Figure 13: Similar users' visual explanation	39
Figure 14: Prediction cycle.....	41
Figure 15: Login page	42
Figure 16: Registration page	42
Figure 17: Home screen after scanned for ring	42
Figure 18: Music screen with genre choosing	42
Figure 19: Workout screen without any training selected	43
Figure 20: Workout screen with training selection	43
Figure 21: Workout screen in the start of training.....	43
Figure 22: Workout screen in the start of training showing data	43
Figure 23: choose training history	44
Figure 24: Training history	44
Figure 25: Viewing the songs information	44
Figure 26: Admin dashboard	44
Figure 27: viewing run statistics.....	45
Figure 28: Colmi R06 Smart ring.....	45
Figure 29: Gantt-1	51
Figure 30: Gantt-2	52
Figure 31: BioLight M70 pulse oximeter and Heart Rate Measurements from the Ring	54
Figure 32: Table of heart rate trial results.	54
Figure 33: Admin dashboard displaying run statistics and scoring events.....	57
Figure 34: Song history with trigger reasons and effectiveness classification.	57
Figure 35: Run events log - Small BPM deviation resulting in a 2-point deduction.....	57
Figure 36: Diviations per phase with vs without music	59
Figure 37: Run score per user with and without music.	60
Figure 38: Run score changes per user between without and with music.	61
Figure 39: Deviation in BPM for every phase of users runs with music.....	61
Figure 40: Deviation in BPM for every phase of users runs without music.....	62

1 INTRODUCTION

For many runners, maintaining the right training intensity is a persistent challenge. Traditional training programs often prescribe fixed paces or durations, but these fail to account for real-time physiological changes such as fatigue, hydration, or environmental conditions. As a result, runners may push themselves too hard, risking injury and burnout, or too little, leading to ineffective training and slower progress. Without continuous feedback, it is difficult for athletes - whether beginners, competitive runners, or rehabilitation patients - to stay within their optimal heart rate zone, which is critical for improving endurance, speed, and overall fitness.

Music has long been shown to influence athletic performance by increasing motivation, focus, and stamina [1] [4]. However, in most cases, music during workouts is static - songs are chosen by the runner before the session and remain unchanged regardless of the runner's performance. This creates a missed opportunity: if the music could adapt dynamically to the runner's physiological state, it could provide both motivation and regulation, helping the athlete sustain optimal performance.

OnBeat explores this space by combining wearable technology, mobile development, and responsive feedback to create a system that adjusts music in real time based on the runner's heart rate. At the core of OnBeat is a smart ring that monitors the user's BPM and performance metrics during workouts. This data is sent via Bluetooth to a mobile app built with React Native, which communicates with a Python backend. The backend processes the data, monitors whether the runner is within their target heart rate zone, and dynamically adjusts the music tempo to encourage the runner to push harder, ease off, or maintain their pace.

By aligning music with physiological signals, OnBeat makes training more engaging and adaptive. It motivates runners at critical moments, provides continuous feedback, and records session data for long-term analysis and improvement. This approach transforms music from a background element into an active training partner - one that helps runners achieve their goals more efficiently and enjoyably.

1.1 Problem Description

Traditional training programs often rely on fixed exercise intensities or durations, neglecting the runner's current physiological state. This static approach overlooks the natural fluctuations in heart rate that can occur during exercise due to factors such as fatigue and real-time heart rate variability. Consequently, runners may push themselves too hard, leading to fatigue or injury, or not hard enough, resulting in less effective training sessions. Without real-time feedback on their heart rate, runners may struggle to stay within their optimal training zones, hindering improvements in endurance, speed, and overall fitness [1] [2].

1.2 Motivation

The motivation behind this project is to enhance the workout experience by leveraging the proven benefits of music on motivation, focus, and endurance. Research has shown that music can significantly influence physical performance and psychological engagement during exercise [1] [2]. However, static playlists often fail to adapt to the user's changing physical state, limiting their effectiveness.

This project aims to address that limitation by incorporating real-time feedback mechanisms that dynamically adjust the tempo of music based on the user's heart rate. A wearable device monitors heart rate and communicates with the app, allowing the system to respond accordingly - for example, by slowing down the music if the heart rate exceeds a desired range, or increasing the tempo to encourage effort and motivation. Additionally, the system learns from historical data and patterns of similar users to further personalize the experience. This adaptive music system seeks to improve efficiency, engagement, and overall workout satisfaction through intelligent, context-aware audio feedback.

1.3 Goals

- Design a personalized music-based training system that dynamically adapts to the runner's physiological state in order to enhance workout outcomes and engagement.
- Keep runners within their optimal heart rate zone and intended training intensity through real-time feedback and adaptive music tempo.
- Provide personalized and flexible training experiences that improve fitness results over time.
- Enhance motivation by selecting and adjusting music tracks based on the user's heart rate and workout phase.

1.4 Overview of the Approach

The OnBeat project introduces an innovative, data-driven approach to enhancing the running experience by synchronizing music with a runner's real-time heart rate. At its core, OnBeat is a wearable-guided fitness system that connects a smart ring to a mobile application, which in turn communicates with a backend server for intelligent analysis. The result is a personalized workout experience that dynamically adapts music to support the runner's pace and heart rate zone.

The system is composed of three main components:

1. **The Smart Ring** - Worn by the user, it continuously measures heart rate and transmits data via Bluetooth.
2. **The Mobile App (React Native for Android)** - Connects to the ring, receives real-time heart rate data, and sends it to the backend. It also includes a built-in music player and a user-friendly interface to manage workout sessions.
3. **The Backend Server (Python)** - Receives biometric and performance data (including real-time running speed), analyzes it, and selects the appropriate music tempo and track to keep the runner in their target heart rate zone or help adjust it up or down.

The system intelligently responds to fluctuations in heart rate: if the runner is below or above the target zone, the app adjusts the music accordingly - either to motivate the runner to increase intensity, slow down, or maintain consistency.

The system also tracks long-term performance metrics and provides session summaries, allowing users to monitor progress and tailor their training further.

1.5 Usage scenarios

In order to explain the ideas behind OnBeat, two scenarios are introduced in the following sections.

1.5.1 Runner Scenario: Personalized Music-Guided Run

Dana, an amateur runner, opens the OnBeat app and wears her smart ring. She selects a fitting training for her and starts running. The ring immediately begins transmitting her real-time heart rate to the app, which in turn communicates with the backend server. Based on her current heart rate and speed, the system selects and plays music with a tempo that matches or slightly pushes her target heart rate range.

As Dana runs, the music dynamically adjusts: if her heart rate drops below the target zone, the tempo increases to motivate her; if she exceeds the target, the app selects a slower track to stabilize her. Throughout the session, Dana focuses on her run while OnBeat takes care of the pacing and motivation through music. At the end of the run, she receives a summary of her performance including BPM trends, pace, and time spent in different heart rate zones.

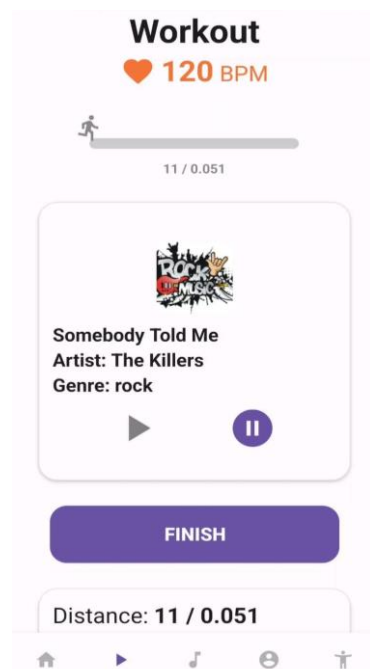


Figure 1: Workout screen with a run

1.5.2 Trainer Scenario: Real-Time Remote Monitoring

Eli, a personal trainer, logs into the OnBeat admin dashboard to monitor his trainees in real time. He sees Dana's session as it unfolds: her current heart rate, speed, and live stats are streamed from the app to the backend and displayed on his screen. Eli can track whether she's staying in the desired training zone and how her performance evolves over time.

With this information, Eli can provide live feedback or analyze the session afterward to suggest adjustments in pacing, intensity, or training focus. This remote visibility allows him to guide multiple runners effectively, even if they are training independently in different locations.

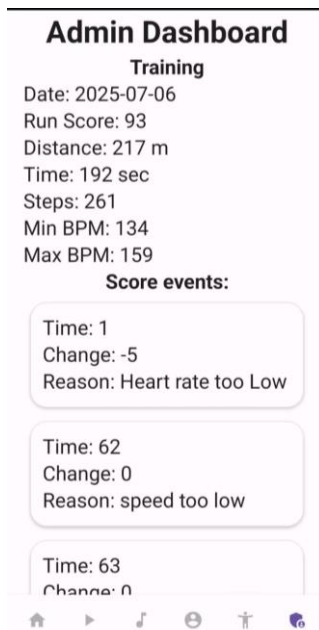


Figure 2: Admin Dashboard with Figure's 1 stats

1.6 Audience

OnBeat is designed for a range of users. Fitness enthusiasts, Beginner runners and Competitive athletes use it to improve endurance, track heart rate zones and maintain a steady pace with real-time feedback through adaptive music. Rehabilitation patients rely on it to stay within safe heart rate ranges. Fitness trainers can use the system both as runners and as observers - they can view associated runners, review training results, and analyze performance data.

1.7 Glossary

The following terms are divided into two categories: generic and OnBeat specific terms.

1.7.1 Generic terminology

- **LightGBM** - A gradient boosting framework that is efficient for large-scale, tabular data. It's used in OnBeat to predict effective songs based on user and session data.
- **Cold Start** - A common problem in recommendation systems where the system lacks historical data for new users or items. OnBeat handles this by using data from similar users.
- **Similar Users** - A machine learning technique where data from users with comparable profiles (age, BMI, goals, favorite genres) is used to make predictions or recommendations for new or inactive users. In OnBeat, this helps personalize sessions even when user data is limited.
- **BLE (Bluetooth Low Energy)** - A power-efficient wireless protocol used to transmit data between devices. OnBeat uses BLE 5.0 to connect the smart ring and mobile app.
- **PPG Sensor** - Photoplethysmography sensor that measures heart rate by detecting blood volume changes in the skin. OnBeat's smart ring uses this for BPM tracking.
- **Real-Time System** - A system that processes input and delivers output with minimal delay. OnBeat adjusts music within seconds based on live heart rate data.

- **Machine Learning Model** - An algorithm trained on data to make predictions or decisions. OnBeat uses ML to recommend songs that enhance performance.
- **Session** - A defined period of user activity - in OnBeat, this refers to a single workout run.
- **Heart Rate Zone** - A range of BPM values tailored to a runner's fitness level, used to guide training intensity.
- **BMI (Body Mass Index)** - A numerical measure calculated from a person's weight and height to assess body composition. OnBeat uses BMI groups to help define training zones and compare users to similar profiles (Overweight, Normal, Underweight).

$$\text{BMI} = \frac{\text{weight (kg)}}{\text{height (m)}^2}$$

Figure 3: BMI formula

1.7.2 OnBeat specific terminology

- **BPM (Beats Per Minute)** - Measures the user's heart rate. OnBeat tracks BPM in real time to adapt music and training feedback.
- **Tempo** - The speed of a music track, which OnBeat adjusts to match the runner's BPM or performance stage.
- **Music Effectiveness** - A score calculated after each run showing how well a song helped maintain the target BPM.
- **Adaptive Music Selection Algorithm** - The logic used by OnBeat to dynamically adjust music based on current heart rate, speed, and user profile.
- **Run Score** - A performance indicator computed during a session based on how closely the runner stays within target zones.
- **Training Plan** - A user-defined set of goals like distance, time, or intensity. OnBeat uses it to guide and evaluate sessions.
- **Session History** - The log of past workouts stored in the system, including BPM trends, song data, and performance results.
- **Genre Preference** - A user setting indicating preferred music genres. OnBeat uses it to personalize song selection.
- **Zone Deviation** - A condition where the runner's BPM is outside the optimal range. OnBeat reacts by adjusting the music.
- **Score** - A numerical value representing the overall quality of a workout, influenced by BPM consistency, tempo adjustments, and music effectiveness.
- **Performance Zones** - Ranges of heart rate values (BPM) established to categorize different levels of performance during training.
- **Optimal Performance Zones** - Specific performance zones identified as most beneficial for the intended training outcome, defined individually for each training.
- **Performance Goals** - objective of guiding the runner into the designated performance zones and sustaining them there throughout the training session.

2 LITERATURE SURVEY

2.1 Problem Survey

The Problem Survey aims to explore the challenges and gaps that have necessitated advancements in the field of wearable technology and fitness applications. The modern world has seen an exponential rise in wearable technology and fitness tracking applications, yet significant gaps remain in how these technologies interact with users to provide meaningful, real-time support during physical activities. Among these challenges, the most pressing issues can be categorized as follows:

1. **Lack of Seamless Synchronization Between Physiological State and External Stimuli:** Music has long been recognized as a powerful motivator in physical activities such as running and exercise [1]. Studies have shown that music with a tempo matching an individual's heart rate or pace can improve endurance, focus, and overall enjoyment [2][5]. However, current fitness applications and devices often fail to adapt to real-time biometric data, leaving users with static playlists that may not align with their activity levels or desired intensity [6]. This disconnect not only reduces the potential benefits of music synchronization but also creates a gap in user satisfaction and engagement [1].
2. **Limited Real-Time Adaptability in Wearable Devices:** Wearable devices like smart rings and fitness trackers have proliferated in recent years, offering features such as heart rate monitoring and activity tracking. Despite their capabilities, most of these devices focus on data collection rather than real-time interaction and actionable insights [3]. Integrating such devices with applications that adapt dynamically to user needs is a challenge that has yet to be fully addressed. Existing solutions are often either overly complex, limited to specific ecosystems, or prohibitively expensive, making them inaccessible to many users [6].
3. **User Experience Challenges in Fitness Applications:** From a user's perspective, the lack of real-time adaptability in fitness applications translates to a suboptimal experience. Runners and fitness enthusiasts frequently face difficulties in finding music that aligns with their rhythm, which can diminish motivation and disrupt their flow [4]. For individuals seeking to optimize their workouts or achieve specific fitness goals, the inability to customize music tempo based on biometric feedback is a missed opportunity [5]. The psychological and physiological benefits of synchronized music remain underutilized due to these shortcomings [1]. Improving the psychological aspect of the trainee involves creating a motivating and personalized experience by synchronizing music tempo with real-time metrics such as heart rate [2]. Upbeat tracks during high-intensity periods can boost energy and focus, while calming music during recovery phases helps manage fatigue [4]. Additionally, real-time positive reinforcement, adaptive feedback, and goal-oriented session summaries further enhance engagement and mental resilience, turning each workout into a rewarding and enjoyable experience [6].
4. **Technical Barriers in Integration and Data Processing:** The integration of wearable devices with mobile applications presents additional challenges. Bluetooth connectivity, device compatibility, and the need for accurate and responsive data analysis are significant hurdles [3]. Many existing systems

struggle to process biometric data in real-time and translate it into actionable changes, such as adjusting music tempo or generating personalized feedback [5]. Moreover, the lack of open-source solutions in this domain limits innovation and accessibility, leaving developers and users dependent on proprietary systems that may not fully meet their needs [6].

2.2 Solution Survey

The Solution Survey explores the existing approaches and innovations that attempt to address the challenges identified in the Problem Survey. Below are solutions corresponding to the specific challenges outlined earlier, along with how our system aims to address them.

- 1. Solutions for Seamless Synchronization Between Physiological State and External Stimuli:** Music streaming services such as Spotify and Apple Fitness+ offer workout playlists tailored to specific activity types and tempos. However, these solutions do not dynamically adapt to real-time biometric data such as heart rate or energy expenditure [1]. Some research prototypes have explored adaptive music algorithms that adjust tempo based on biometric feedback, but they remain in early development and are not yet widely available in consumer applications [6].
Our Approach: Our system integrates real-time biometric monitoring from wearable devices - specifically the Colmi R06 smart ring used in this project - and adjusts music tempo dynamically based on the user's heart rate [5]. This provides a continuously responsive experience, helping maintain motivation and engagement during physical activity [2].
- 2. Solutions Addressing Limited Real-Time Adaptability in Wearable Devices:** Wearable devices like Apple Watch, Garmin fitness trackers, and Oura rings have gained popularity for monitoring biometric data, including heart rate, sleep patterns, and calorie expenditure. While these devices excel at data collection, they do not offer seamless integration with music applications that adapt in real time [3]. Many operate within closed ecosystems, restricting third-party integration and broader usability. Some fitness bands offer limited third-party connectivity, but interoperability remains a challenge [6].
Our Approach: Our system is built as an open-source platform that integrates with Bluetooth-enabled biometric devices. By focusing on interoperability and flexibility, we enable real-time music adaptation based on biometric input - avoiding the limitations of closed hardware ecosystems [4].
- 3. Solutions for Enhancing User Experience in Fitness Applications:** Applications such as RockMyRun and TempoRun have attempted to match music tempo to a user's pace or activity level. However, their adaptability is often limited to predefined song categories or static algorithms rather than true real-time biometric control [1]. Some experimental platforms explore dynamic tempo adjustment, but these solutions are still rare in mainstream applications [5].
Our Approach: Rather than relying on predefined logic, our system uses real-time algorithmic adjustments to match music tempo with the user's heart rate [6]. This creates a more personalized and motivating experience, continuously adapting to the user's effort level throughout the session [2].

4. **Solutions Addressing Technical Barriers in Integration and Data Processing:** Several open-source initiatives and research efforts have attempted to build frameworks for music adaptation driven by biometric input. Proprietary platforms like Fitbit and Apple Health offer some integration capabilities but remain locked to their ecosystems [3]. Research into Bluetooth communication and real-time data processing has improved, but challenges remain in building fully responsive systems that adjust playback in real time without perceptible delay [5].
Our Approach: We use low-latency Bluetooth communication and efficient real-time data handling to achieve responsive behavior [4]. Our system's server-based architecture enables rapid biometric processing and tempo decisions, minimizing delays and improving the overall music–runner synchronization experience [6].

2.3 Discussions and Conclusions

The Discussion and Conclusion section evaluates how our system, OnBeat, effectively addresses the challenges identified in the Problem Survey and improves upon existing solutions discussed in the Solution Survey. By integrating real-time heart rate monitoring with adaptive music playback, our system provides a dynamic and user-centric approach to enhancing workout experiences. Below, we analyze how our approach directly resolves the identified challenges and contributes to advancements in wearable fitness technology.

1. **Addressing the Lack of Seamless Synchronization Between Physiological State and External Stimuli:** Existing solutions such as Spotify and Apple Fitness+ offer workout playlists with fixed tempos but fail to adapt dynamically to real-time biometric feedback [1]. Research prototypes have attempted to create adaptive music algorithms, but they have not been widely implemented in commercial products [6].
Our System's Implementation: we implemented real-time heart rate tracking using the Colmi R06 smart ring and dynamic music tempo control based on heart rate changes [5]. This enables users to maintain a consistent training pace with music that adapts naturally to their physical effort. The system ensures synchronization between exertion and auditory feedback, improving motivation and focus throughout the workout [2].
2. **Enhancing Real-Time Adaptability in Wearable Devices:** Many wearable fitness devices, such as Apple Watch and Garmin trackers, collect biometric data but do not offer real-time interaction with external applications for music adaptation [3]. Their closed ecosystems limit integration with third-party solutions, restricting users' ability to customize their workout experiences [6].
Our System's Implementation: OnBeat was developed specifically to work with the Colmi R06 smart ring, which we reverse-engineered to enable real-time heart rate tracking via Bluetooth [4]. While other commercial wearables often restrict third-party integration, our custom implementation allowed full control over the R06's data stream, enabling seamless music synchronization based on live heart rate input [5].
3. **Improving User Experience in Fitness Applications:** Applications like RockMyRun and TempoRun have attempted to match music tempo with a user's activity level, but their algorithms are limited and often do not make full use of biometric feedback [1]. This results in imprecise tempo matching and reduced workout engagement [5].

Our System's Implementation: Our system uses real-time logic and data modeling techniques - such as LightGBM regression and similar-user profiling - to determine optimal music tempo based on the runner's current heart rate [6]. This allows for personalized, responsive music experiences that adapt continuously to the user's physiological state [2].

4. **Overcoming Technical Barriers in Integration and Data Processing:** Current fitness solutions face common obstacles, including Bluetooth connectivity issues, limited device compatibility, and high-latency data processing [3]. Platforms like Fitbit and Apple Health provide tracking capabilities but operate within closed ecosystems that limit external integration [6].

Our System's Implementation: OnBeat uses low-latency Bluetooth communication, efficient real-time biometric data analysis, and a server-based architecture to ensure near real-time responsiveness [4]. Our implementation also involved reverse-engineering the Colmi R06 communication protocol, allowing us to bypass ecosystem restrictions and create a fully custom integration. These choices help ensure continuous music feedback with minimal delay and high reliability [5].

Conclusion:

The OnBeat system successfully addresses key issues in current wearable fitness technology by combining real-time heart rate monitoring, logic-driven music adaptation, and open-platform integration [1]. By bridging the gap between biometric feedback and dynamic user interaction, OnBeat elevates the workout experience beyond what most commercial solutions currently offer [2].

Moreover, this project contributes to ongoing progress in health technology, human-computer interaction, and real-time adaptive systems [3]. By analyzing heart rate data and adjusting music in real time through learned user profiles and modeling techniques, OnBeat shows how biometric-driven feedback can enhance both performance and user satisfaction [4]. This innovation supports not only individual fitness enthusiasts but also lays a foundation for future work in adaptive interfaces and personalized fitness systems [6].

3 TECHNOLOGICAL SURVEY

3.1 SQL vs NO SQL

SQL databases store data in structured, relational tables with predefined schemas, supporting powerful queries, joins, and ACID transactions. They are ideal when data integrity, relationships, and complex querying are important.

NoSQL databases offer flexible, schema-less data models - such as documents or key-value pairs - and are designed for horizontal scalability and handling very large volumes of simple reads/writes, but often sacrifice strong transactional guarantees.

For OnBeat, PostgreSQL - a SQL database - was chosen because the system's data is highly relational: users, trainers, sessions, songs, song-play events, and effectiveness scores all link together. ACID compliance ensures that live session events and music effectiveness data are recorded without loss or duplication, which is crucial for accuracy. PostgreSQL's advanced querying capabilities, including joins, aggregates, and window functions, allow efficient analysis of BPM trends, session summaries, and song performance directly in the database. Its JSONB support also offers flexibility for storing semi-structured metadata when needed, blending some NoSQL benefits with SQL reliability.

Additionally, PostgreSQL provides strong indexing, performance tuning, and a mature ecosystem, making it easy to integrate with OnBeat's Python backend and React Native app. NoSQL's primary advantages - extreme horizontal scaling and schema flexibility - were less critical for OnBeat's current scale, while PostgreSQL's integrity, analytical power, and low-latency transactional consistency aligned perfectly with the project's real-time and historical analysis requirements.

3.2 Algorithm: LightGBM

Several algorithms were evaluated for OnBeat's adaptive music selection. **K-Nearest Neighbors (KNN)** was simple and interpretable but too slow and memory-heavy for real-time use. **Decision trees** and **random forests** handled mixed data well but were less suited for ranking and could be computationally heavy. **Neural networks** offered flexibility but required large datasets, had higher latency, and were unnecessary for the current scale.

LightGBM was ultimately chosen for its speed, low-latency predictions, ability to handle mixed and missing data, and support for ranking candidate songs. By incorporating similar users' data, it also mitigates cold start issues, making it both efficient and well-suited for OnBeat's real-time requirements.

3.3 Wearable Technology: Colmi R06 Smart Ring

The OnBeat system is built around the **Colmi R06 smart ring**, a compact wearable device designed for continuous biometric monitoring. The R06 combines convenience and functionality by embedding a suite of health sensors within a lightweight and discrete form factor. It was chosen for this project due to its affordability, accessibility, and potential for reverse engineering via Bluetooth Low Energy (BLE) communication.

Key features of the Colmi R06 relevant to the project include:

- **Heart Rate Monitoring:**
The ring uses a PPG (photoplethysmography) sensor to measure heart rate in real time. This data serves as the primary input for music adaptation in the OnBeat system.
- **Bluetooth 5.0 Connectivity:**
The R06 transmits biometric data using BLE, which allowed us to establish a custom connection between the ring and the OnBeat mobile app. Reverse engineering of the BLE characteristics and services enabled us to read heart rate data without relying on the official app.
- **Compact Form Factor:**
Weighing only 4.6g and made of durable materials like aluminum alloy and ceramics, the R06 is ideal for long-term wear during running or physical activity. Its non-intrusive design improves user comfort and encourages consistent usage.

The choice to use the Colmi R06 demonstrates how affordable, commercially available devices can be repurposed for advanced, real-time applications in health and fitness. It also highlights the role of **reverse engineering** and **open-source development** in expanding the capabilities of closed systems.

3.4 Python

Python was selected for the implementation of the backend server and algorithmic components due to its versatility, readability, and extensive ecosystem of **libraries** for **data processing** and **machine learning**. Python's rich set of scientific and signal processing libraries, including NumPy and Pandas, allows for effective implementation of the adaptive music tempo algorithm. This ensures that the backend can perform real-time analysis, adapt music playback according to user performance, and maintain low latency in communication with the mobile application.

3.5 React native

React Native was chosen for the development of the OnBeat mobile application. Its **component-based architecture** and **hot-reload capabilities** streamline the implementation of the user interface, enabling efficient prototyping and refinement of features such as real-time metric displays, workout program management, and music preference selection. Furthermore, React Native's ability to integrate with native device features, including **Bluetooth connectivity** and **local storage**, facilitates seamless communication with the smart ring and supports offline music playback through local caching.

4 REQUIREMENTS AND SPECIFICATION

Functional requirements of the project are the necessary conditions that define the core behavior and operations of the system. **Non-functional requirements**, on the other hand, are concerned with the quality attributes and constraints that influence how the system delivers its functionality.

4.1 OnBeat functional requirements

The following requirements define the primary functional capabilities of OnBeat:

1. **Running Program Selection:** The system will allow users to view and select running programs with specific optimal performance zones, providing flexibility to customize their training routines.
2. **Workout Selection:** The system will allow users to select or receive recommended workouts tailored to their weight and age group, ensuring that training sessions are appropriate for their fitness level.
3. **Preference Selection:** Users will be able to specify their running preferences, such as outdoor runs, treadmill sessions, or indoor track workouts, ensuring context-aware functionality.
4. **Real-Time Monitoring:** During running sessions, the system will monitor and display key metrics including heart rate (BPM), speed, and distance.
5. **Adaptive Music Tempo:** If the user's performance deviates significantly from the set optimal performance zones, the system will dynamically adjust the music tempo to help regulate pace and maintain performance goals.
6. **Music Genre Selection:** Users will be able to select their preferred music genres, allowing for personalization of the listening experience during runs.
7. **Run History and Analytics:** The system will maintain a record of past running sessions, providing access to summaries that include metrics such as distance, average BPM, and average speed.
8. **Learning and Adaptation:** OnBeat will include a learning feature that adapts music playback over time by analyzing historical performance and behavior data, improving music selection for future sessions.
9. **Accessibility Features:** Users will be able to adjust text size and interface colors within the application without loss of content or functionality, supporting diverse user needs.
10. **Data and Communication:** The system will establish a secure connection with the smart ring to receive heart rate data and synchronize it with the application, while also integrating music playback capabilities for local caching.

4.2 OnBeat non-Functional requirements

The following requirements define system constraints and quality attributes:

1. **Performance:** The system must transmit heart rate data from the smart ring to the mobile application with minimal latency (under 3 seconds), and music tempo adjustments should occur within 1 second to ensure seamless synchronization with the user's activity.
2. **Usability:** The mobile application should provide an intuitive and user-friendly interface, enabling effortless control of running sessions, real-time metric viewing, and access to performance history.
3. **Scalability:** The backend server must support multiple concurrent users without performance degradation, ensuring a reliable experience for all users.
4. **Availability:** The system must provide uninterrupted music playback through local song caching.
5. **Security:** Application-Server communications must occur via secure, encrypted connections, profile access must be restricted through authenticated login methods.
6. **Compatibility:** The application must support recent versions of Android and be compatible with specific Bluetooth-enabled smart rings, as well as music formats such as MP3 and AAC.
7. **Portability:** The application should be lightweight and optimized for mobile devices, ensuring minimal battery consumption during extended sessions.

4.3 Specification - The Scope of the OnBeat Project

After careful consideration of the requirements and available resources, the following scope was defined:

Inside the Scope of the Project:

- Full implementation of all listed functional requirements, including program management, workout selection, adaptive music tempo, real-time heart rate monitoring, and performance analytics.
- Integration of music playback with support for local caching.
- Support for secure communications with the backend server.
- Implementation of all non-functional requirements, including performance targets, usability, security, and portability.

Outside the Scope of the Project:

- Integration with third-party fitness platforms (e.g., Strava or Google Fit) is not included in the initial scope.
- Advanced route planning or GPS navigation features beyond simple workout selection.
- Support for operating systems other than Android.
- Predictive analytics beyond the initial learning feature (e.g., AI-driven training recommendations).

5 THE ARCHITECTURE

To describe the architecture of the OnBeat application, we begin by outlining the data requirements, including its sources and the transformations it undergoes. Next, we explain the roles involved in interacting with the application. Finally, we describe the activities performed during system use, leading to an overview of the system life cycle - the stages the system passes through during operation.

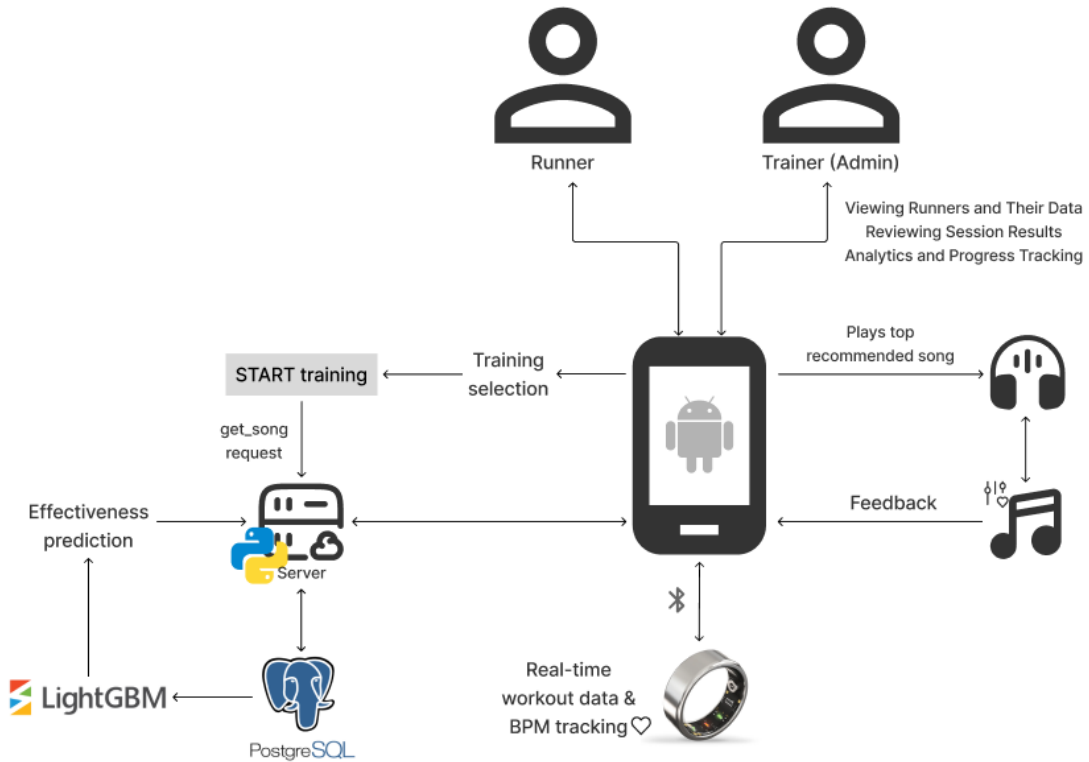


Figure 4: System architecture

5.1 OnBeat Data

The OnBeat system is built around three primary data sources: **physiological sensor data**, **user profile data**, and **music metadata**. Together, these provide the foundation for real-time training feedback, adaptive song selection, and long-term personalization.

- **Physiological Sensor Data-**

The main live input comes from the COLMI R06 Smart Ring, which integrates two key sensors. The VCare VC30F photoplethysmography (PPG) sensor measures heart rate at a sampling rate of 1 Hz with an accuracy of $\pm 2-5$ BPM. This data is critical for detecting whether the runner is within, above, or below their optimal training zone in real time. In parallel, the STK8321 3-axis accelerometer provides step counts, estimates distance, and classifies movement states (e.g., walking, running, stationary). Together, these measurements allow the system not only to track effort but also to distinguish between different types of activity - ensuring that music adaptation is triggered by genuine running performance rather than unrelated heart rate spikes.

- **User Profile and Historical Data-**

Each runner has a stored profile that includes demographic and physiological attributes such as age, BMI, and baseline heart rate ranges. The system also records workout histories, including session summaries (distance, duration, average BPM, speed) and past

interactions with music. This data allows the system to tailor training targets to individual needs and detect trends over time. For example, if a runner consistently struggles to stay in the aerobic zone during intervals, OnBeat can adjust its feedback and music cues to encourage better pacing in future runs.

- **Music Metadata and Effectiveness Scores-**

Music plays a dual role in OnBeat: as a motivator and as a regulatory tool. Each song in the curated library is tagged with tempo (BPM), genre, artist, and play count. More importantly, the system attaches an effectiveness score to each track, updated after every workout. These scores capture how well a song helped the runner stay in their target zone - whether it boosted performance, maintained it, or proved ineffective. Over time, this creates a personalized music dataset where the system learns which genres and tempos work best for each runner at different workout stages (beginning, middle, end).

- **Data Storage and Machine Learning Integration-**

All persistent data is managed in a PostgreSQL database, which stores user profiles, workout histories, and music metadata with their effectiveness labels. During workouts, live sensor data streams to the React Native mobile app via Bluetooth Low Energy (BLE 5.0), and from there to the backend for analysis. For machine learning, OnBeat combines the runner's personal history with data from similar users (matched by features such as age, BMI, and pace) to address cold-start problems. This dataset feeds into a LightGBM model, which scores candidate songs filtered by BPM, genre, and play history, ensuring that recommendations are both physiologically relevant and motivationally effective.

5.1.1 The internal and external representations

Internal Representation:

The OnBeat system manages several internal datasets that store user profiles, workout history, heart rate logs, and music metadata. These datasets are stored in a PostgreSQL database and accessed through the Python backend. Data is structured in relational tables with rows representing individual records (e.g., sessions, songs, or BPM readings) and columns representing attributes (e.g., genre, tempo, run score, effectiveness).

Incoming real-time data from the smart ring - such as heart rate, step count, and distance - is processed by the backend. This raw sensor data is stored both in its original form and in a processed form after applying validation and cleaning steps, including filtering noisy readings, normalizing BPM ranges, and classifying activity phases (begin, middle, end). The cleaned and aggregated data feeds into the adaptive music selection algorithm.

Historical datasets are also used for model training. The LightGBM model is periodically retrained using stored session results, song effectiveness scores, and similar users' data. This enables the system to improve song recommendations and adapt to user preferences over time.

External Representation:

The processed data is presented to users through the OnBeat mobile application, built in React Native. The user interface includes interactive screens such as the workout process screen (showing real-time BPM, pace, and current music), post-run summaries (distance, heart rate trends, and run scores), and history views for previous sessions.

Trainers and administrators access aggregated data through specialized views, which display associated runners, performance analytics, and training progress. The application's external representation focuses on clarity - using simple color indicators (green, yellow, red) for heart rate zones, charts for BPM trends, and structured lists for historical sessions. This ensures that both runners and trainers can easily interpret results and adjust training accordingly.

5.1.2 Initial start-up data

Initialization Procedures:

The initialization of the OnBeat system ensures that the application, backend, and wearable device function cohesively from the first run. The process involves the following key steps:

1. **Database Setup:** Initial datasets are loaded into the PostgreSQL database, including default music metadata (title, artist, tempo, genre, and effectiveness placeholders) and default user profiles for testing. These records provide a foundation for the recommendation system and music selection logic.
2. **Preloading Music Cache:** A predefined set of music tracks (500) is cached locally in the application for offline operation. These tracks include a range of tempos for testing the adaptive playback feature even without network connectivity.
3. **Initial Model Preparation:** The LightGBM model is initialized with training data from predefined similar user groups and historical sample sessions. This provides a starting point for recommendations before enough user-specific data is collected.
4. **System Preprocessing at Start-Up:** During startup, any initial cleaning or format validation is applied to imported datasets (e.g., ensuring music metadata is normalized, removing duplicate entries, validating BPM ranges). This ensures the system starts with reliable baseline data for analysis and real-time adjustments.

5.1.3 View persistence

All user sessions, training plans, and performance data are stored in the PostgreSQL database, enabling users to access their history at any time. A runner can resume a workout that was paused, review detailed statistics of previous runs, or compare progress over time. Trainers can also access the training history of their associated runners, review past results, and track improvements.

The application maintains persistent views of:

- **Workout History:** Complete list of past runs, including date, distance, BPM trends, and run scores.
- **Music Effectiveness:** Logged performance impact of songs played, enabling analysis of which tracks work best.
- **Trainer Dashboards:** Persistent access to assigned runners' progress and analytics.

5.2 The OnBeat approach

5.2.1 Data Collection & Preprocessing

Based on the commonly used formula: **Maximum Heart Rate (MHR) = 220 - Age**

Age	Light (50-60%)	Moderate (60-70%)	Hard (70-85%)
20	100-120 BPM	120-140 BPM	140-170 BPM
30	95-114 BPM	114-133 BPM	133-162 BPM
40	90-108 BPM	108-126 BPM	126-153 BPM
50	85-102 BPM	102-119 BPM	119-145 BPM
60	80-96 BPM	96-112 BPM	112-136 BPM
70	75-90 BPM	90-105 BPM	105-128 BPM

Figure 5 - BPM age table

- Heart Rate: Sampled every second, accurate to $\pm 2-5$ BPM in most conditions.
- Steps & Distance: Used to differentiate real running from incidental BPM increases (e.g., stress).
- Workout Metadata: Phase (start, middle, end), pace, elapsed time, and target zones.
- Song Metadata: Tempo (BPM), genre, and historical effectiveness.
- BMI Categorization: Users are assigned to BMI groups:
 - Underweight: $BMI < 18.5$
 - Normal: $18.5 \leq BMI < 25$
 - Overweight: $25 \leq BMI$

This affects target BPM zones and recommended intensity.

5.2.2 Song Effectiveness & Scoring

Each song played during a workout is assigned an effectiveness score based on its impact:

- Effective = -1, Effectiveness = Worsened. If the song is finished or replaced and the BPM and performance were within range at the beginning and were not within range at the end, then it is a deterioration and will receive a score of -1
- Effective = 0, Effectiveness = Maintained bad. If the song is finished or replaced and the BPM and performance were not within range at the beginning and were not within range at the end, then it remains bad and is not considered a deterioration, and therefore receives a score of 0.
- Effective = 1, Effectiveness = Maintained good. If the song is finished and the BPM and performance were within range at the beginning and at the end, then it remains good and is not considered an improvement, and therefore it receives a score of 1.
- Effective = 2, Effectiveness = Improved. When the song is finished and there was an improvement in the BPM and the user's performance, it receives a score of 2.

In the end, each time the song is played, the score of each song is added to the score it had before, and this will be the weighted score. High-effectiveness songs are favored in future recommendations. Low or negative scores push the song lower in selection priority.

5.2.3 Machine Learning Model

Algorithm LightGBM:

- Handles mixed features: BPM, user profile, BMI group, genres, song metadata.
- Efficient on large datasets.
- Handles missing data well (critical for new users).
- Data Split: 80% for training, 20% for testing.
- Cold Start Solution: If a new user has insufficient history, the system uses “similar users”:
 - Matches on BMI group, age range, preferred genres and workout goal.
 - Pulls top-scoring songs from those users to bootstrap recommendations.

5.2.4 Song Selection Process

When a new song request is triggered:

- Input Parameters:
 - Current BPM
 - Preferred genres
 - Songs already played in this session (no repeats)
 - Phase of workout (begin, middle, end)
- Candidate Filtering:
 - Select songs with tempo in $\text{'BPM} \pm 5'$.
 - Filter by preferred genres.
 - Remove already-played songs.
- Scoring:
 - ML-predicted score from LightGBM.
 - Adjust score using historical effectiveness weighting.
- Selection:
 - Pick the highest-scoring song.
 - Send to the app for immediate playback.

5.2.5 Real-Time Adaptation

During playback:

- Tempo Adjustments: Made instantly via `TrackPlayer` speed control.
- Song Change Rule: If 40 seconds pass and the BPM is still outside the target zone, switch to a different song better suited to the needed adjustment.
- If BPM too low:
 - Increase playback speed (e.g., $\times 1.2$, $\times 1.4$).
 - Encourage higher pace with more upbeat music.
 - If after 40 seconds it seems that the song is not raising the heart rate to the target zone, we will switch to a song whose tempo is 10 times higher.
- If BPM too high:
 - Reduce playback speed (e.g., $\times 0.8$, $\times 0.6$).
 - Switch to calmer tracks to aid recovery.
 - If after 40 seconds it seems that the song is not lowering the heart rate to the target zone, we will switch to a song whose tempo is 10 times lower.

5.2.6 Workout Phases

Previous studies have shown that music tempo preference varies across different stages of exercise [5][6], and we also consulted a fitness professional to align these phases with practical training approaches

- **Begin Phase:** High-energy, motivational tracks to raise BPM to target zone.
- **Middle Phase:** Tempo-matched tracks to keep BPM stable within the optimal range.
- **End Phase:** Gradual tempo decreases to help recovery while maintaining engagement.

5.2.7 Continuous Learning

After each workout:

- Effectiveness scores for all songs are updated.
- Session data is stored in DB for retraining.
- Model retrains periodically, incorporating new results to improve recommendations.

5.3 OnBeat Processes

Before a run begins, the user opens the app, selects a workout plan by time or distance, and chooses music preferences. The app connects to the smart ring over Bluetooth, verifies live heart-rate streaming, and initializes the workout screen and session timers so everything is ready the moment the user starts.

Once the user taps Start, the session enters live monitoring. The app continuously displays core metrics - heart rate and elapsed time - while background services keep the Bluetooth link stable. Internally, the system maintains a lightweight session state such as the current phase, timers, and which track is playing, so it can react smoothly to what's happening in real time.

Music playback begins immediately to provide uninterrupted guidance. The player adapts during the session to align with the current workout context and phase, and the app avoids obvious repetitions within a single workout to keep the experience fresh and engaging. All of this happens without the user needing to manage a queue or tap extra controls.

The workout flows through three phases. The Start phase eases the user into the effort, the Middle phase focuses on maintaining a steady rhythm, and the End phase helps the user cool down. Transitions are driven by the plan and the user's progress, so the experience feels natural rather than abrupt.

Throughout the run, the app provides in-session feedback that is clear but not intrusive. Visual indicators reflect status and progress, and subtle cues keep the runner engaged without overwhelming them with information or interruptions.

When the user finishes the workout, playback stops and device streaming is closed cleanly. The app presents a concise summary showing duration and basic trends from the session, and the user can optionally add notes or subjective impressions before saving.

Past sessions remain easy to explore. A history view lists previous runs with their dates and headline metrics, while a detailed view shows the phase timeline and the sequence of tracks. Simple charts and tables make it straightforward to identify trends over time without requiring advanced analysis.

Behind the scenes, the app logs essential session artifacts - metadata, timeline events, and the track sequence - and stores them securely. This record powers summaries, supports future insights, and can be made available for exports if needed and permitted.

Recommendation quality improves over time. New sessions are incorporated in the background, and updates roll out without interrupting the user. The impact is visible in subsequent workouts through smoother choices and a more tailored feel.

All components coordinate to keep latency low and the experience smooth. The mobile app, wearable connection, backend services, and storage communicate in an event-driven manner, with resilient messaging and local caching to ensure playback continues even when connectivity is limited.

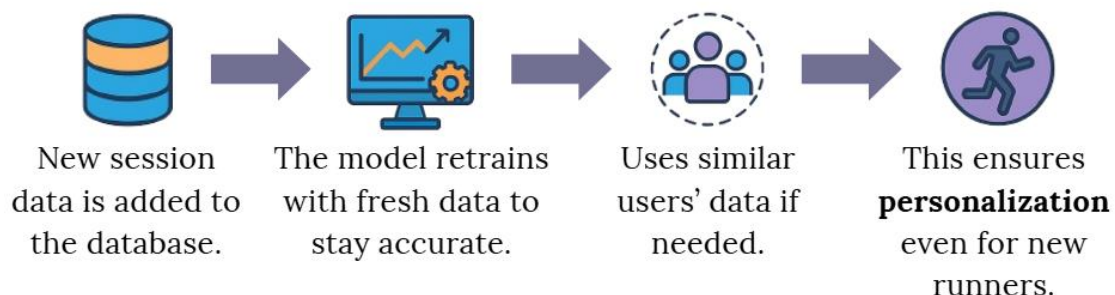


Figure 6: Continuous Improvement

Finally, the system is designed to recover gracefully from real-world hiccups. Temporary device disconnects trigger quick reconnection attempts, and if online sources are unavailable the app falls back to local content. Clear, brief messages inform the user of what's happening without disrupting the run.

5.4 Roles

OnBeat defines two primary roles: the Runner and the Trainer. The Runner is the main end-user, responsible for using the application during workouts. They can start, pause, and stop training sessions, view real-time BPM, pace, and song details, and benefit from adaptive music that adjusts based on their heart rate zone. Runners can also review their workout history, analyze performance data, and set personal training plans and genre preferences. The Trainer role extends these capabilities by enabling the trainer to act as both a Runner and a coach. Trainers can view all associated runners, track their progress, analyze trends in performance, and review workout histories.

5.5 The life-cycle view of a system

This diagram represents the OnBeat prediction and feedback life cycle:

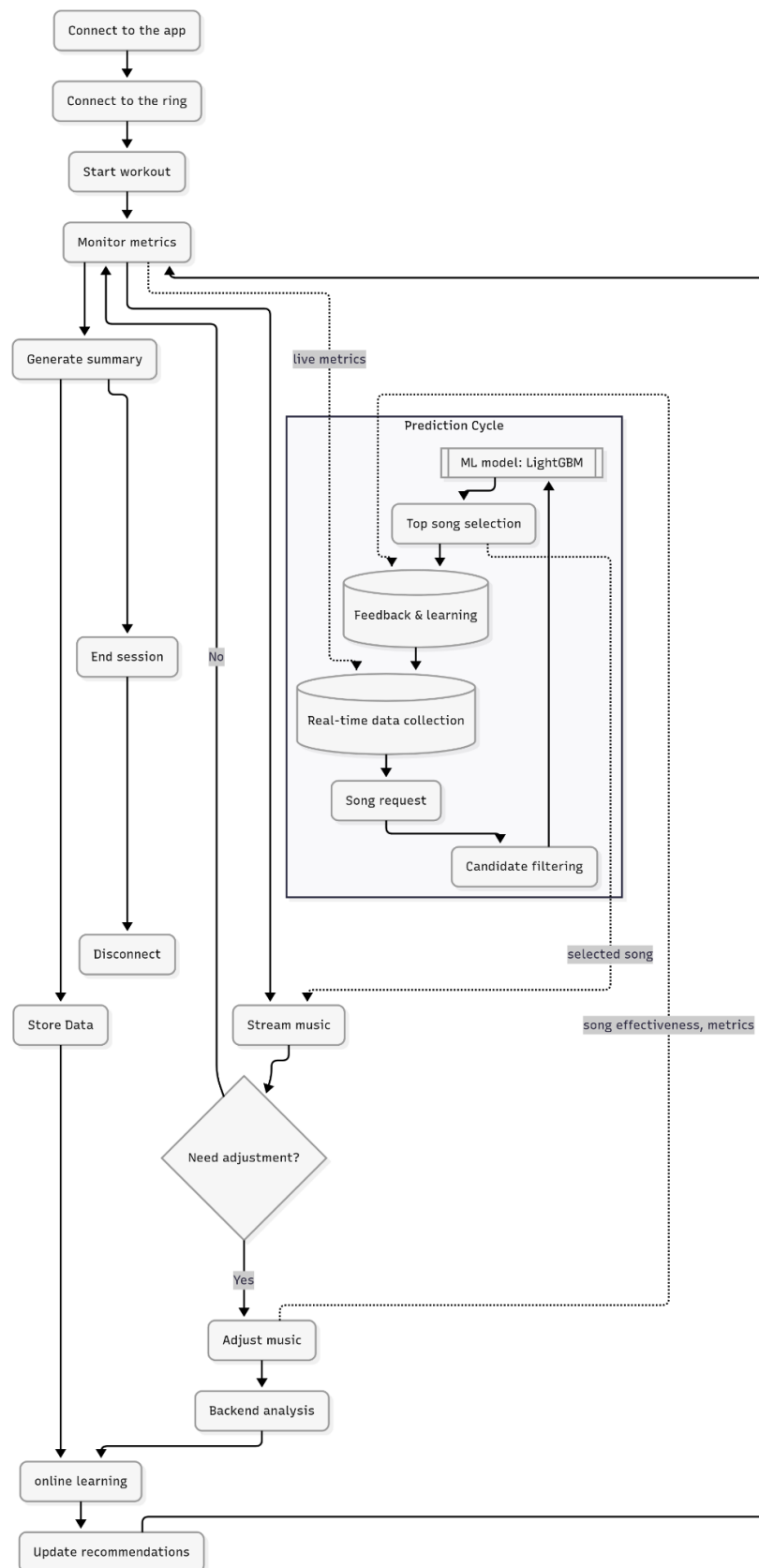


Figure 7: Life-cycle diagram

The life cycle begins when the user connects to the OnBeat mobile application, pairs the smart ring, and starts a workout. From that moment, the system continuously monitors real-time performance metrics such as heart rate, steps, distance, and speed. These live measurements are streamed both to the mobile app for immediate feedback and to the backend for analysis.

While the user is running, the system streams a track adapted to the runner's condition, but before making changes it checks whether adjustment is necessary. If no change is needed, the current track continues. If an adjustment is required, the application modifies the playback speed or switches the song, ensuring the runner remains in the optimal heart rate zone. Each of these decisions is logged and analyzed by the backend to refine the system's future responses.

The inner loop is the prediction cycle. Real-time data collected from the ring is used to generate a song request. Candidate songs are filtered based on tempo proximity and preferences, and then scored using a LightGBM machine learning model. The top-ranked song is selected and returned to the mobile app, where it begins playing. Afterward, the system captures the effectiveness of this choice, records its impact on the runner's performance, and feeds this information into a learning loop. Over time, the model retraines with new data and applies fallback strategies, such as drawing insights from similar users, to provide increasingly accurate recommendations.

When the workout ends, the system generates a summary for the user, displaying analytics such as distance covered, and song effectiveness. The session then closes, disconnecting the ring, while all metrics and results are stored in the database. This stored data fuels online learning in the backend, enabling the algorithm to update recommendations and adapt to the runner's personal history. In this way, each new workout benefits from the insights of previous ones, creating a continuous improvement cycle where the music becomes more aligned with the runner's needs and goals.

6 SYSTEM DESIGN

This section discusses the design of the OnBeat system, detailing how its architecture and core elements are mapped to the components that implement and operate them. An early design decision was to structure the system around a mobile application communicating with a Python - based backend via WebSockets for real-time data exchange. This approach was chosen to support continuous heart rate monitoring from the Colmi R06 smart ring and to enable adaptive music playback with minimal latency. Certain features considered during the initial planning phase - such as integration with multiple wearable devices - were excluded from the final design and are therefore not covered in the requirement and specification sections.

6.1 Data components

This part describes the system's data entities, including user profiles, run histories, biometric readings, and music metadata, along with their storage and relationships in the database.

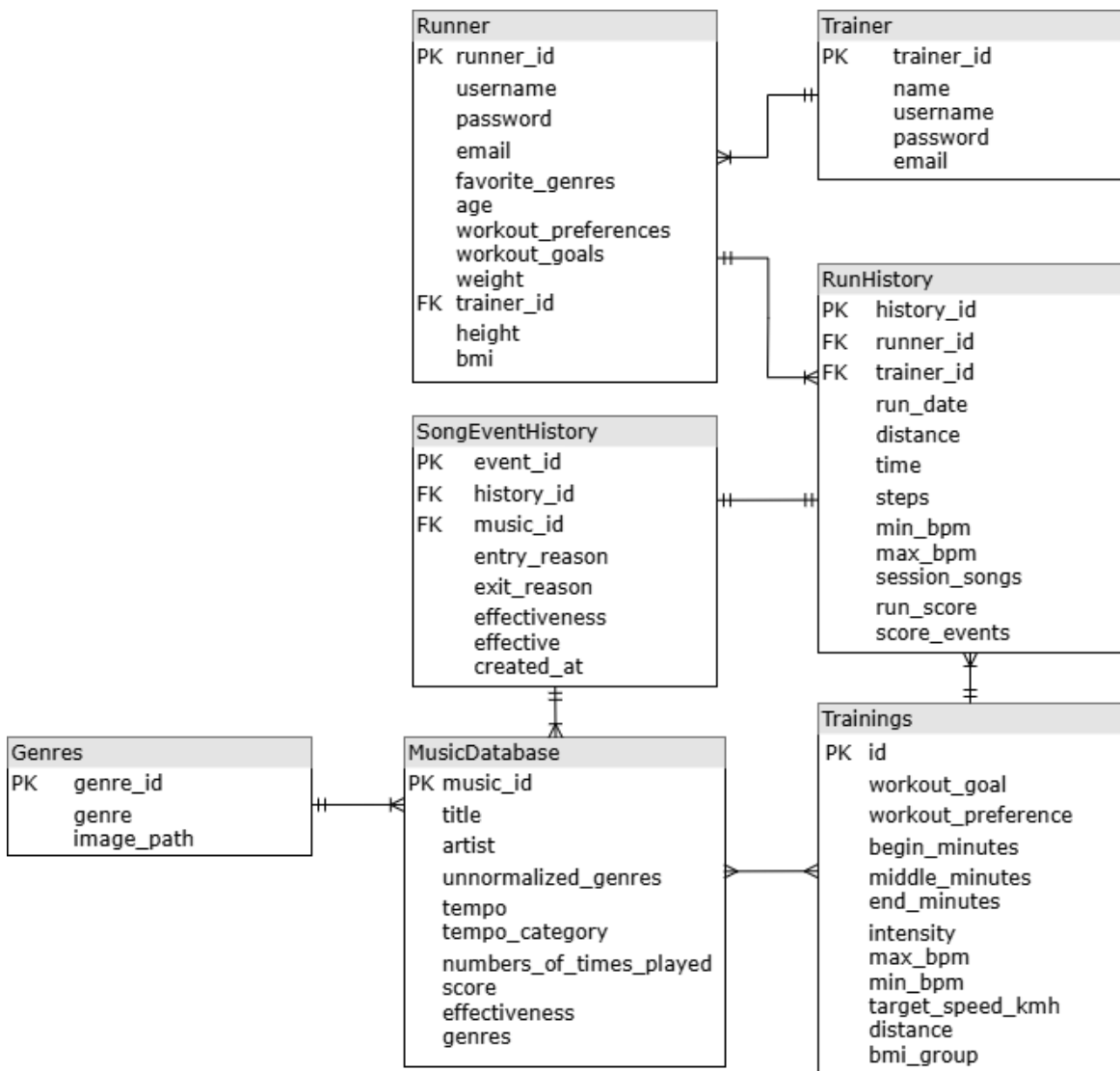


Figure 8: Data Model Diagram

Runner - Stores information about each user of the system, including personal details, workout preferences, and links to their trainer.

Trainer - Contains details of trainers who can manage and monitor multiple runners.

RunHistory - Records individual workout sessions, including distance, time, steps, heart rate ranges, and run-specific statistics.

SongEventHistory - Logs when and why specific songs are played or stopped during a workout, along with their effectiveness.

MusicDatabase - Contains the catalog of songs available to the system, including metadata such as title, artist, tempo, and play statistics.

Genres - Lists music genres and associated images for classification and user preference matching.

Trainings - Defines workout templates and goals, including time segments, intensity, and target ranges for heart rate and speed.

6.2 Process components

1. Bluetooth Data Acquisition

This process manages the connection between the OnBeat mobile application and the Colmi R06 smart ring. It is responsible for discovering the device, establishing a secure Bluetooth connection, and continuously receiving real-time heart rate, step count, and distance data. Reliable data acquisition ensures accurate monitoring throughout the workout session.

2. Training Plan Evaluation

The training plan evaluation process compares the processed sensor data against predefined workout targets. These targets are based on the user's selected training plan, which specifies heart rate ranges, speed goals, and workout phase durations. To define realistic and effective plans, we researched these workout structures in consultation with actual fitness professionals. Unlike traditional training systems that adjust workout intensity directly, our system responds by adapting the music playback to either maintain, increase, or decrease the user's perceived intensity, aligning the workout experience with the intended training goals.

3. Music Selection Algorithm

The music selection algorithm is the core logic that matches current workout conditions to the most suitable song. It ranks available tracks based on tempo proximity, genre preferences, and past effectiveness, while also applying rules to prevent frequent song changes. The algorithm ensures music playback supports the runner in maintaining or reaching the desired heart rate range.

4. Playback Control

Playback control is responsible for delivering music to the user seamlessly. The song title is chosen by the algorithm running on the server and sent to the app, where the track is already stored locally and played without delay. If playback speed needs to be adjusted due to high or low BPM, this adjustment happens directly in the app to ensure real-time responsiveness. When a song switch is required, the instruction also comes from the algorithm. In addition, the system logs each playback event, including which song was played, any speed adjustments, and the outcome of the session, enabling further analysis of music effectiveness.

5. Run Logging and Storage

Run logging records all relevant workout details, including biometric trends, selected songs, performance deviations, and run summaries. This information is stored in the database for future analysis, allowing both runners and trainers to review performance data.

6. Real-Time Synchronization

Real-time synchronization handles the continuous exchange of data between the mobile

application and the backend server via WebSockets. This enables trainers to monitor runner performance live and provides instant updates on song selections and workout metrics.

7. **Results and Analytics Display**

This process retrieves and presents post-run summaries, historical performance trends, and song effectiveness statistics to the user. The results are displayed in a clear, accessible format to support performance review and training adjustments.

8. **Error Handling and Logging**

Error handling ensures that any issues - such as connection drops, sensor faults, or data processing errors - are detected and managed without interrupting the workout. Logging captures key events and errors for debugging and system monitoring.

6.3 **Communication components (user-computer interaction)**

Mobile Application Interface

The mobile application provides the primary point of interaction for the runner. It enables the user to connect to the Colmi R06 smart ring, start and stop workout sessions, and view live metrics such as heart rate, steps, and speed. The interface is optimized for clarity during physical activity, with large, easily readable elements and minimal distraction.

Trainer Dashboard Interface

The trainer dashboard is a web-based interface that allows trainers to monitor runners' live workout data. It displays heart rate, pace, workout phase, and current song selection in real time. This enables trainers to make informed coaching decisions and provide guidance during the session.

Real-Time Feedback Mechanism

Real-time feedback is delivered through visual indicators, color-coded heart rate zones, and dynamic status updates. Runners receive immediate cues when their heart rate is above or below target, helping them adjust effort without needing to stop or check detailed statistics.

Workout Summary and Analytics View

After a session, the user can review a detailed workout summary. This includes biometric statistics, run score, phase breakdown, and song effectiveness. The analytics view supports performance tracking over time and helps identify trends in training progress.

Settings and Customization

Users can configure their workout preferences, such as target heart rate range, preferred music genres, and intensity levels. Trainers can assign training plans and adjust session parameters to match each runner's goals.

6.4 **Interactions**

This section will present different diagrams in order to describe the OnBeat design.

6.4.1 **Activity diagram**

Flow of activities across roles - Runner, Smart Ring, Android App, Backend (Python), ML Model (LightGBM), and Database (PostgreSQL).

The Runner starts a workout in the Android app and wears the Smart Ring, which immediately streams real-time signals (heart rate each second, plus motion like steps/speed) over BLE. The Android App receives these readings, shows live metrics, and, when music is enabled, builds a song request containing the current BPM, the runner's selected genres, and IDs of tracks already played this session to avoid repeats. That request goes to the Backend via WebSocket.

On the server, the Backend does candidate filtering: it pulls songs from the Database that match the runner's genres and fall within a narrow tempo/BPM window (e.g., ± 5) while excluding repeats. The filtered set is handed to the ML Model (LightGBM), which predicts how effective each candidate will be right now, using context like current BPM, workout stage (begin/middle/end), user history, and similar-users fallback for cold starts. The model returns a score per candidate; the backend responds with the top song and its metadata.

Back on the phone, the Android App loads the returned track into the player and starts playback. During the feedback loop, the app keeps monitoring heart rate and pace: if the runner is below target, it can nudge tempo up; if above target, it can slow the track or switch to something calmer. Throughout the session the app logs events (BPM, stage, which song played, why it started/stopped, playback-rate changes) and labels each song's effectiveness (improved/maintained/worsened). These events are written back to the Database.

When the Runner finishes, the app stops the music, shows the post-run summary (distance, tempo accuracy, song list with effectiveness), and persists the session. Periodically, the Backend performs continuous improvement: it retrain the LightGBM model on newly collected sessions, so the next workout starts with better-personalized predictions even for edge cases (e.g., new users, new genres).

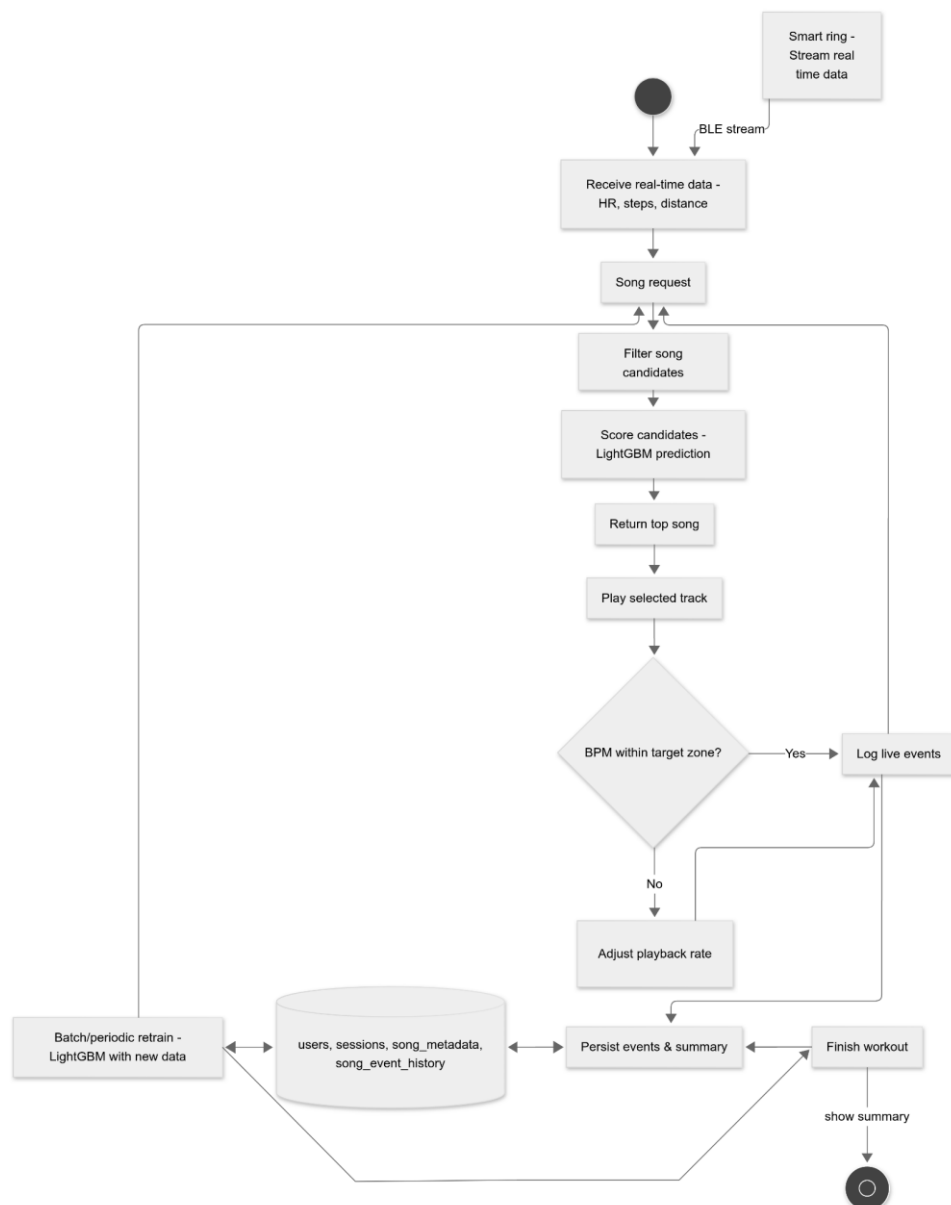


Figure 9: Activity diagram

6.4.2 Design sequence diagram

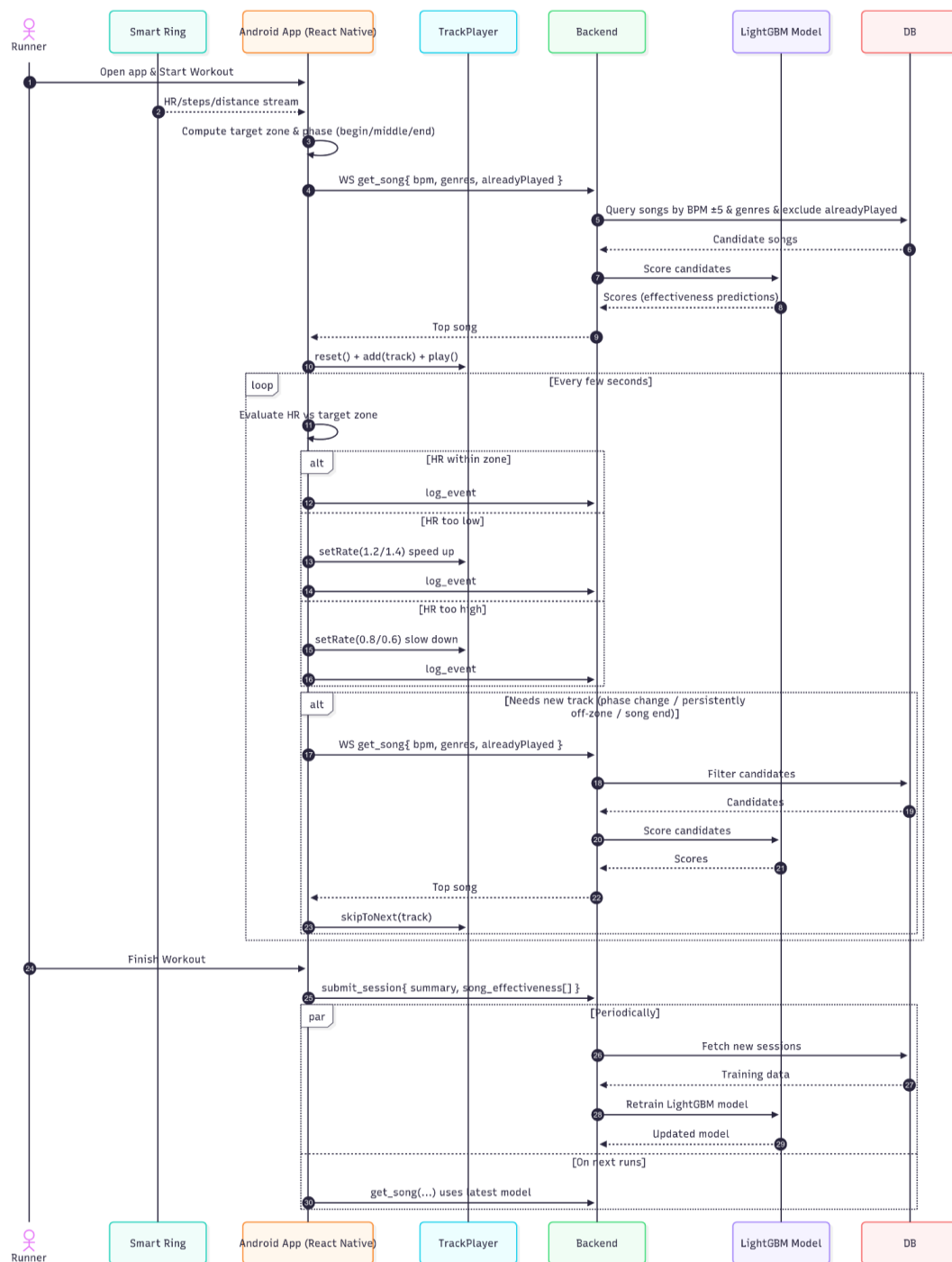


Figure 10: Sequence diagram

The OnBeat sequence diagram shows how the system's components interact during a workout session to deliver personalized, real-time music adaptation. The process begins when the runner starts a workout on the Android app, which connects to the smart ring via Bluetooth. The ring streams live heart rate, step count, and movement data to the app, which updates the user interface and sends song requests to the backend through a WebSocket connection. Each request includes the runner's current BPM, workout phase, preferred genres, and songs already played in the session. The backend queries the database to filter suitable candidates by BPM range, genre, and non-repetition, then passes these candidates to the LightGBM model for scoring based on user

history, workout phase, and similar-user data. The top-scoring song is returned to the app along with any tempo adjustment instructions. The app loads the track into the music player, plays it, and dynamically adjusts playback speed according to the runner's heart rate. This feedback loop continues throughout the session, logging song effectiveness scores and workout events. At the end of the workout, the app stops music, disconnects the ring, and sends a complete session summary to the backend for storage. Over time, the LightGBM model is retrained with the accumulated session data, improving future song recommendations and ensuring the system becomes increasingly personalized.

6.4.3 Use case diagram

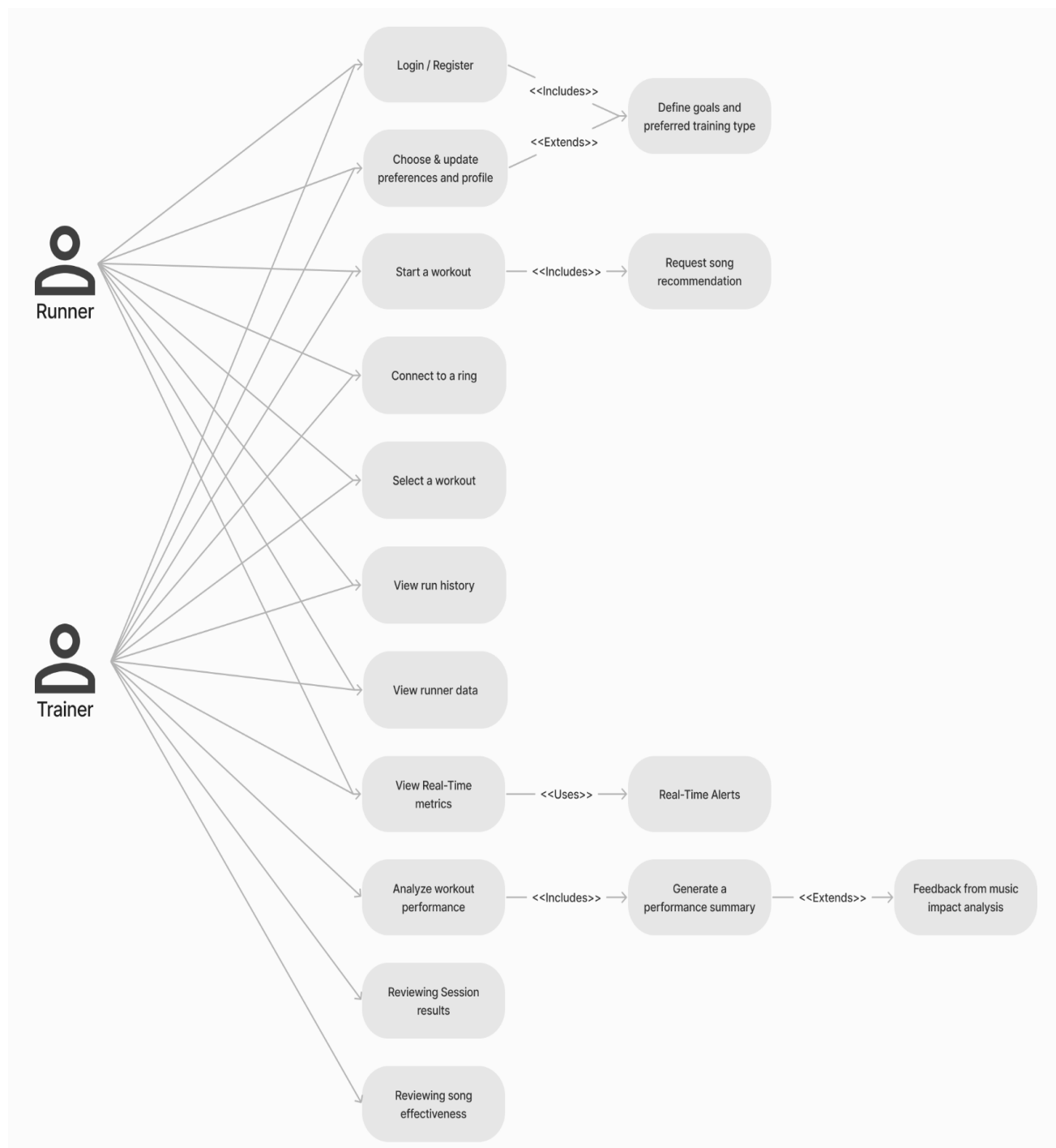


Figure 11: Use-case diagram

6.5 System Architecture

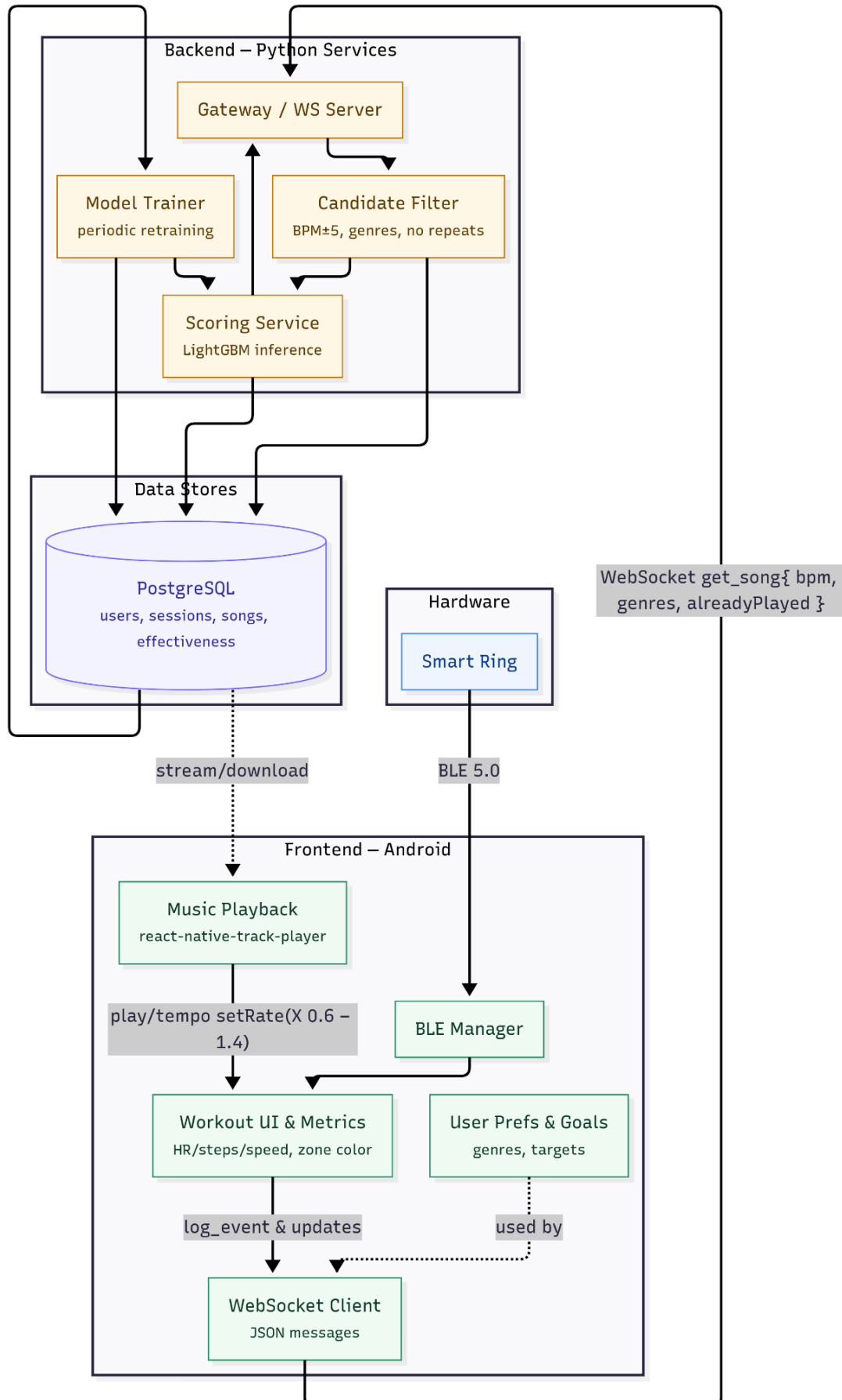


Figure 12: System architecture

This diagram represents the OnBeat system architecture, showing how its hardware, software, and data components interact to deliver personalized, real-time workout adaptation.

At the hardware layer, the smart ring continuously measures the runner's heart rate, steps, speed, and distance. These readings are transmitted via Bluetooth Low Energy (BLE 5.0) to the Android mobile app built in React Native.

The mobile application layer handles three main roles:

1. Live data acquisition from the ring.
2. User interface & workout control, showing metrics and workout phases.
3. Music playback and tempo control via `'react-native-track-player'`.

The communication layer uses WebSocket connection to the backend for data exchange and song requests. The backend layer (Python-based) processes these requests through a candidate filtering module ($\text{BPM} \pm 5$, genre match, no repeats) and the LightGBM machine learning model, which scores songs for effectiveness based on the runner's current state, history, and similar-user data.

The data layer is powered by PostgreSQL, storing user profiles, workout history, song metadata, and logged events such as tempo adjustments and effectiveness scores. Periodically, the backend runs a model retraining service using the accumulated session data, ensuring predictions improve over time.

7 IMPLEMENTATION

The implementation of OnBeat integrates hardware, software, and machine learning components into a unified real-time system. The smart ring streams physiological data via BLE to the Android mobile app, which manages live metrics, workout phases, and adaptive music playback using react-native-track-player. A persistent WebSocket connection to the backend, developed in Python, enables song requests and feedback. The backend filters candidates, applies the LightGBM model to predict song effectiveness, and returns the top recommendation. All data - including user profiles, workout sessions, and song effectiveness - is stored in PostgreSQL, supporting retraining pipelines that continuously improve the model.

7.1 Algorithm Design

The OnBeat system relies on a multi-stage algorithm to adapt music in real time based on the runner's physiological state and workout context. The design combines rule-based filtering, machine learning prediction, and real-time playback adaptation to ensure music is both engaging and effective for performance.

1. Candidate Filtering

When the app requests a new song, the backend queries the PostgreSQL database for suitable candidates. Songs are filtered by:

- **Tempo range:** ± 5 BPM from the runner's current heart rate.
- **Genre preference:** restricted to the user's chosen genres.
- **Non-repetition:** excludes songs already played in the session.

This guarantees a narrowed set of musically coherent candidates, avoiding abrupt BPM mismatches or repeated tracks.

```
#song_selector.py

bpm_lower, bpm_upper = bpm-5, bpm+5

query = """SELECT music_id,title,tempo,genre,artist,score

          FROM musicdatabase

          WHERE tempo BETWEEN $1 AND $2"""

songs = await db.fetch(query, bpm_lower, bpm_upper)

candidates_df = pd.DataFrame([dict(s) for s in songs])
```

2. Machine Learning Scoring (LightGBM)

Filtered songs are passed to the LightGBM model, trained on historical workout sessions. Features include:

- Runner state (current BPM, deviation from target zone, workout phase).
- Song metadata (tempo, genre, score).

- Effectiveness history (was the song motivating or not in prior runs).
- Similar-user fallback (to handle cold-start situations).

LightGBM outputs a probability of effectiveness for each candidate. This approach was chosen for its speed, accuracy with tabular data, and robustness to missing values.

```
#recommend.py

model = lgb.Booster(model_file="ml_model/song_recommendation_model.txt")

ml_df = pd.get_dummies(candidates_df)

for col in feature_cols: #align schema
    if col not in ml_df.columns:
        ml_df[col] = 0

preds = model.predict(ml_df[feature_cols])

ml_df["prob_effective"] = preds

top = ml_df.sort_values("prob_effective", ascending=False).head(top_n)
```

3. similar-user fallback

OnBeat learns per user from song effectiveness logs. But early on (new user) or in niche contexts (rare genre, unusual BPM), there may be too little history to trust a personalized model. The fallback borrows signals from nearby runners - people who behaved similarly - so recommendations are still sensible and on-target.

Typical triggers:

- **Cold start:** user has $< N$ effective song events.
- **Sparse context:** enough total history, but no history in the current BPM/genre bucket.
- **Realtime safety:** the ML score distribution is flat/low (no confident pick).

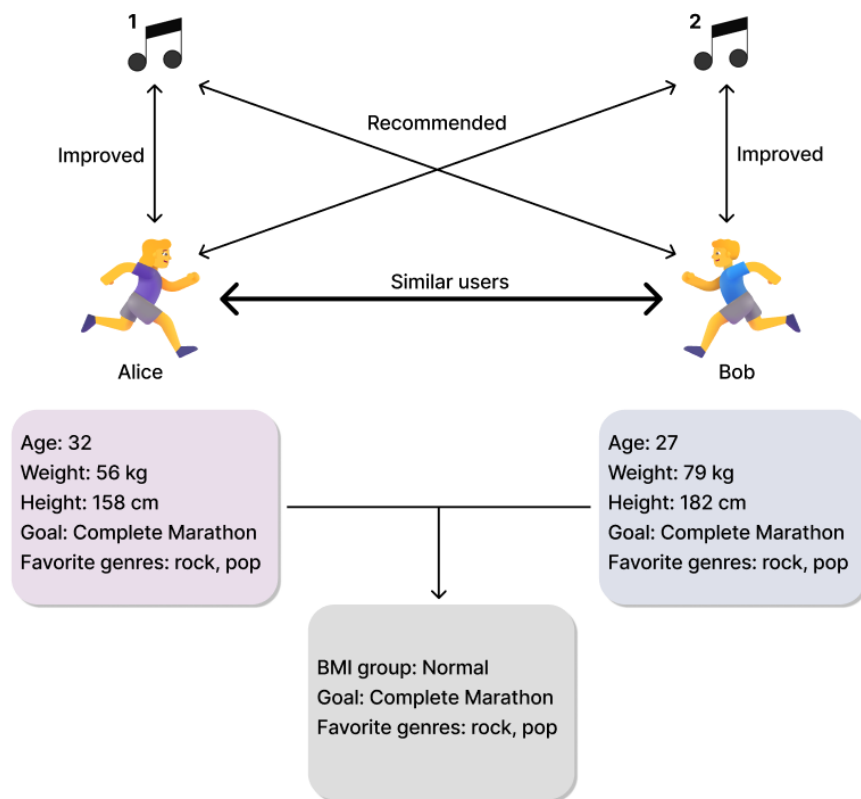


Figure 13: Similar users' visual explanation

4. Inference + ranking (top-N recommendation)

At runtime, the saved model is loaded, the candidate set is aligned to the training feature space (missing one-hot columns are added as zeros), and each song receives a probability of being effective right now.

The top-N are returned. If all predictions tie, a light shuffle preserves variety.

```
#recommend.py

model = lgb.Booster(model_file="ml_model/song_recommendation_model.txt")

ml_df = pd.get_dummies(candidates_df)

for col in feature_cols: #align to training schema
    if col not in ml_df.columns: ml_df[col] = 0

preds = model.predict(ml_df[feature_cols]) #effectiveness probability
ml_df["prob_effective"] = preds

top = ml_df.sort_values("prob_effective", ascending=False).head(top_n)
```

5. End-to-end selection over WebSocket

The app requests a song, the backend filters → scores → returns top recommendations. robust fallbacks send “no suitable song found” if any step yields an empty set.

```
#song_selector.py

top_songs = recommend_song(candidates_df, top_n=5)

await websocket.send_str(json.dumps({"type": "selected_song",
                                     "message": {"recommended_songs":
top_songs}}))
```

6. Playback Adaptation

Once the best song is selected, the app uses react-native-track-player to play it. If the runner's BPM deviates from the target zone:

- Music speed is increased $\times 1.2$ / $\times 1.4$ if HR is slightly / too low (to push intensity).
- Music speed is reduced $\times 0.8$ / $\times 0.6$ if HR is slightly / too high (to encourage recovery).

These tempo adjustments act as real-time biofeedback, helping the runner stay within their optimal zone.

```
#WorkoutProcessScreen.tsx

if (heartRate < min - 10) {
  console.log("HR too low, boosting speed by 1.4.");
  await TrackPlayer.setRate(1.4);
} else if (heartRate < min) {
  console.log("HR slightly low, boosting speed by 1.2.");
  await TrackPlayer.setRate(1.2);
} else if (heartRate > max + 10) {
  console.log("HR too high, slowing speed by 0.6.");
  await TrackPlayer.setRate(0.6);
} else if (heartRate > max) {
  console.log("HR slightly high, slowing speed by 0.8.");
  await TrackPlayer.setRate(0.8);
} else if (heartRate >= min && heartRate <= max) {
  console.log("HR normal, playing at normal speed.");
  await TrackPlayer.setRate(1.0);
}
```

7. Effectiveness Logging & Continuous Improvement

For every track played, the system logs:

- Entry/exit reasons (why the song started/stopped).
- HR deviation events and tempo adjustments.
- Effectiveness label: improved / maintained_good / worsened / maintained_bad.

This data is stored in PostgreSQL and used in periodic retraining of the LightGBM model, ensuring personalization improves over time, As you can see in figure 5.



Figure 14: Prediction cycle

7.2 Interfaces

The interfaces of the OnBeat system are presented primarily through a mobile application developed in React Native, designed for Android devices. The user interface focuses on simplicity and real-time responsiveness, ensuring that both runners and trainers can easily interact with the system during live workouts.

The combination of these interfaces ensures seamless communication between the wearable device, mobile app, backend services, and database, while presenting runners and trainers with a user-friendly, interactive platform for adaptive music-driven training.

7.2.1 Human-computer interfaces (GUI)

Runner Interface: The runner accesses the app to register, log in, and configure their workout preferences such as favorite genres and training goals. During a workout, the interface displays live physiological data (heart rate, steps, distance) streamed from the Colmi R06 smart ring. Visual indicators, dynamic BPM zones, and interactive controls (start, pause, finish workout) provide real-time feedback. The integrated music player allows the runner to monitor the current song and observe tempo adjustments applied by the algorithm.

Trainer/Admin Interface: Trainers can log in to view the profiles and workout histories of their assigned runners. The interface includes dashboards listing session results, effectiveness scores of songs, and progress trends over time. Trainers can update user workout goals, review session logs, and monitor performance metrics collected during training.

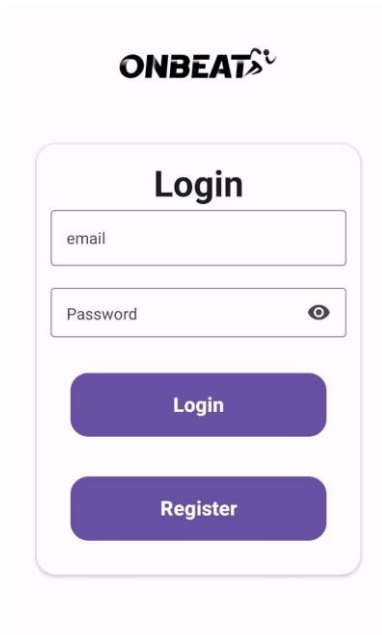


Figure 15: Login page

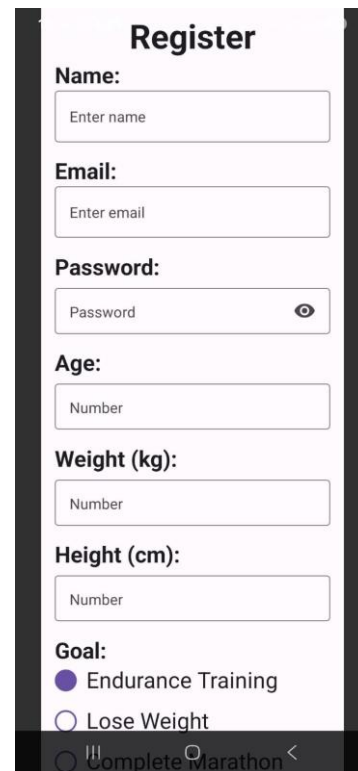


Figure 16: Registration page

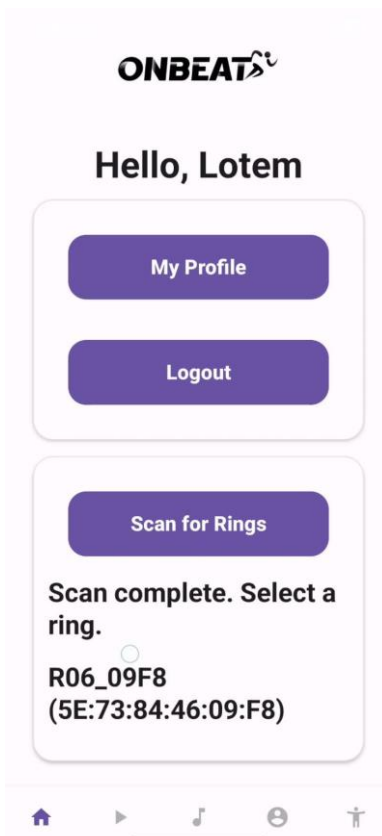


Figure 17: Home screen after scanned for ring

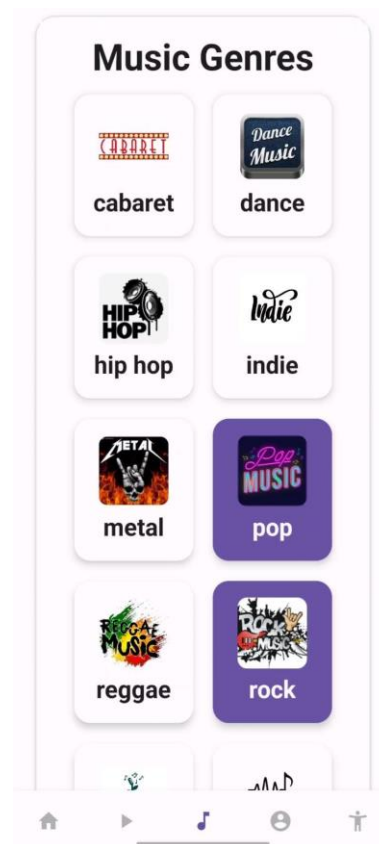


Figure 18: Music screen with genre choosing

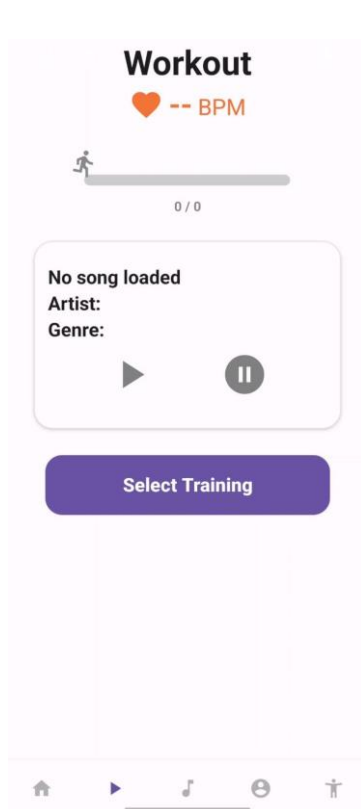


Figure 19: Workout screen without any training selected

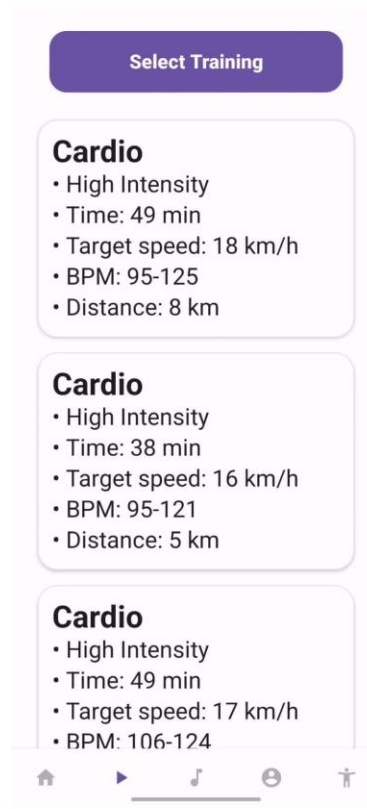


Figure 20: Workout screen with training selection

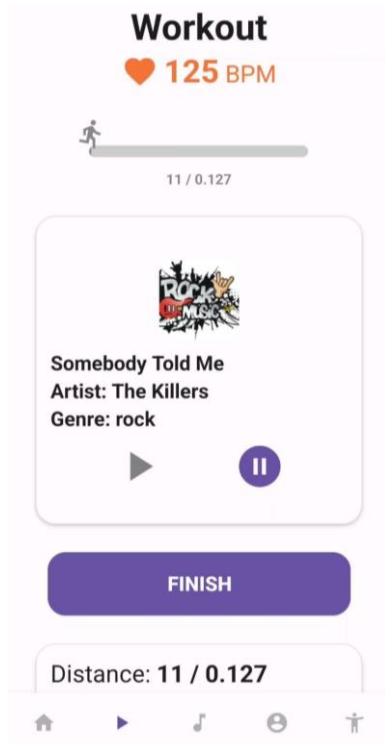


Figure 21: Workout screen in the start of training

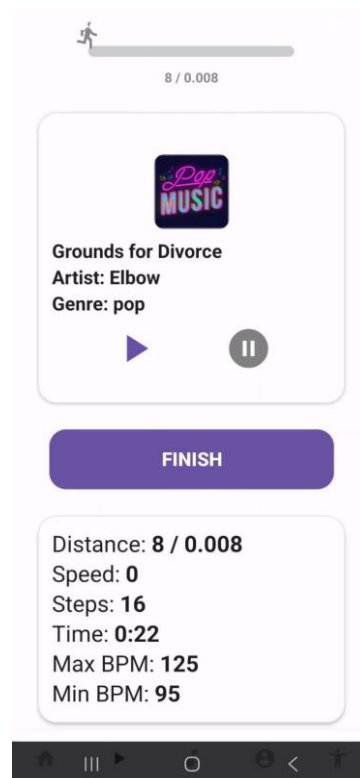


Figure 22: Workout screen in the start of training showing data

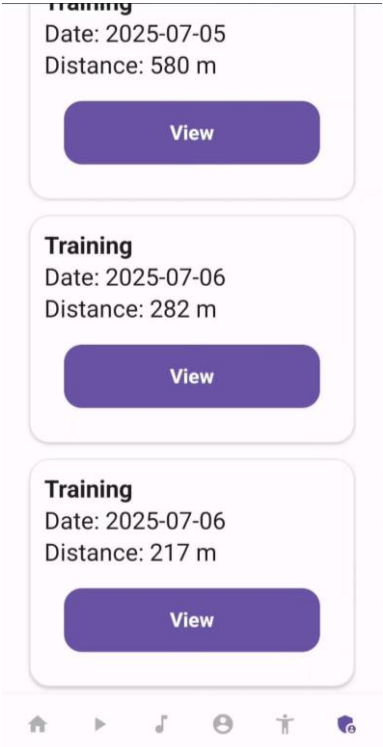


Figure 23: choose training history

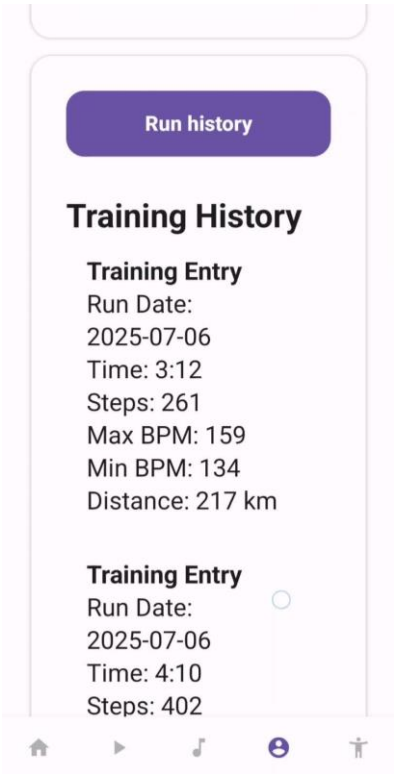


Figure 24: Training history

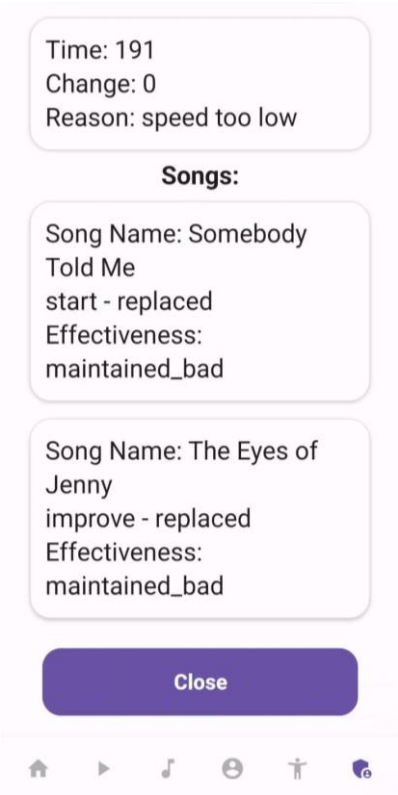


Figure 25: Viewing the songs information

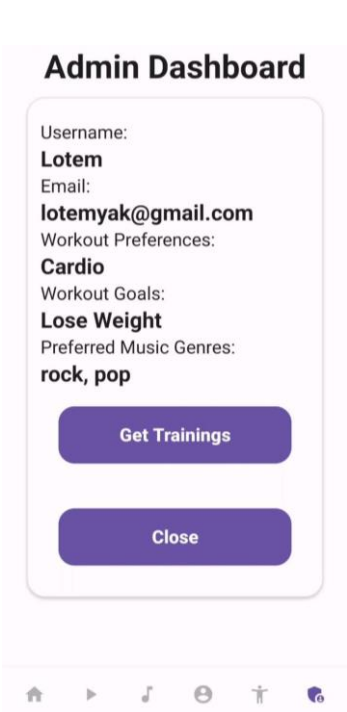


Figure 26: Admin dashboard

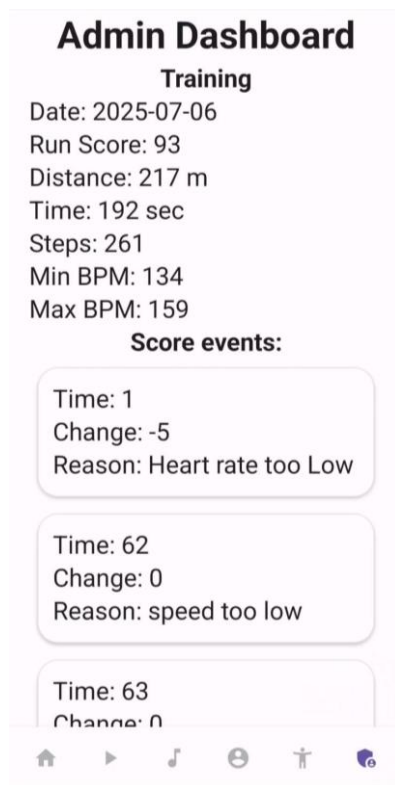


Figure 27: viewing run statistics

7.2.2 The OnBeat-system configuration

The OnBeat system is configured as a layered architecture that integrates hardware, application logic, machine learning services, and data storage, enabling real-time adaptive workout experiences.

1. Hardware Layer

- **Colmi R06 Smart Ring:** Equipped with a VCare VC30F PPG sensor for heart rate monitoring and an STK8321 accelerometer for motion tracking. The device transmits data at 1 Hz via Bluetooth Low Energy (BLE 5.0).



Figure 28: Colmi R06 Smart ring

- **Runner Device:** Android smartphones serve as the host for the OnBeat mobile application, managing ring connectivity, live metrics, and music playback.

2. Application Layer (Mobile App)

Implemented in React Native using Node.js LTS 22.x and JDK 17.

Handles BLE communication with the smart ring, a WebSocket client for backend interaction, and the react-native-track-player module for adaptive playback.

Provides the runner interface (live metrics, workout control, music playback) and the trainer interface (history viewing, progress dashboards).

3. Backend & Machine Learning Layer

Developed in **Python**, deployed as modular services:

- **WebSocket API** for low-latency communication.
- **Candidate Filtering Service** to select songs within ± 5 BPM and user-selected genres.
- **LightGBM Scoring Service** to rank songs based on effectiveness predictions.
- **Telemetry Ingestion Service** to log workout events, HR deviations, and song effectiveness.

The Model Training Pipeline retrains LightGBM periodically using historical logs, ensuring recommendations adapt as new data is collected.

4. Data Layer

- **PostgreSQL** serves as the primary relational database, storing:
 - User profiles, preferences, and training goals.
 - Workout sessions, metrics, and phase data.
 - Song metadata, playback history, and effectiveness scores.
- **Local Song Cache:** Music files are pre-downloaded into the app's `assets/music/` directory to ensure playback without network dependency.
- Optional external object storage/CDN can be integrated for large-scale song distribution.

5. Configuration Parameters

- **Tempo tolerance:** ± 5 BPM around the runner's current heart rate.
- **Playback adjustment range:** $\times 0.6 - \times 1.4$, depending on HR deviation from the target zone.
- **Cold-start fallback:** similar-user profiles are used if the runner's history is insufficient.
- **Retraining frequency:** the LightGBM model is updated periodically using aggregated session logs.

7.3 Development environment

The development of OnBeat was carried out using Visual Studio Code, chosen for its strong support of React Native, Node.js, and Python.

The mobile application was built using React Native with Node.js LTS 22.x, npm 10.9, and JDK 17, providing a consistent Android build pipeline. Android Studio and its emulator were used for integration testing alongside physical devices. The backend services, written in Python, were tested locally in VS Code's integrated terminal and later connected to the PostgreSQL database managed via pgAdmin and hosted on Render.

7.4 Programming languages

Based on the selected technologies, the OnBeat system is divided into three main components, each implemented with specific languages and libraries:

- **Client-side (Mobile Application)** - implemented in TypeScript using React Native.
 - `react-native-track-player` for music playback and tempo control.
 - Bluetooth Low Energy (BLE) modules for smart ring communication.
 - WebSocket client for real-time backend communication.
- **Backend Services** - implemented in Python.
 - **LightGBM** for machine learning-based song effectiveness prediction.
 - **Pandas** and **NumPy** for data preprocessing.
 - **psycopg2** for PostgreSQL integration.
 - **WebSockets** for low-latency message handling.
- **Database** - implemented in SQL using PostgreSQL.
 - Stores user profiles, workout sessions, physiological data, music metadata, and song effectiveness logs.
 - Ensures structured, relational data management for both training pipelines and live session queries.

7.5 Limitations of the system

While OnBeat provides real-time adaptive music and workout guidance, several limitations of the current implementation should be acknowledged:

- **Fixed Selection of Songs:** The system currently relies on a predefined library of songs with annotated BPM values. This restricts user choice and personalization, as users cannot freely integrate their own music collections. Future versions may allow custom library expansions or uploaded tracks to enhance the users' music experience.
- **Predefined Workouts:** The workout programs available in the system are limited to a fixed set of templates designed during development. While these provide structure and consistency, they limit flexibility for users seeking highly customized training plans.
- **Device Dependency:** The system has been validated using the Colmi R06 smart ring, and although designed for extensibility, compatibility with other wearable sensors has not yet been fully tested.
- **Scalability Constraints:** While the backend is architecturally scalable, current testing has been limited to a small number of users due to hardware availability. This restricts validation under large-scale usage scenarios.

7.6 Installation and configuration issues

During deployment and configuration of the OnBeat system, several issues were identified that may impact usability and scalability:

- **Large Application Size:** The app maintains a local cache to ensure smooth real-time operation. While this improves performance and reliability, it increases the overall application size, which may be problematic for users with limited device storage.
- **Scalability Constraints:** The backend scalability has not been fully validated under large-scale usage. Current deployment has only been tested with a limited number of concurrent users. This may present challenges if the system is widely adopted without further optimization and distributed resource management.
- **Device-Specific Variability:** Installation and configuration steps may vary slightly across Android versions and hardware models. While compatibility testing has been conducted on multiple devices, variations in Bluetooth stacks and task handling across Android versions may lead to inconsistent setup experiences for some users.
- **Dependency on External Permissions:** The application requires permissions for Bluetooth connectivity, background processing, and storage access. Incorrectly configured permissions during installation may prevent the system from operating as intended, requiring users to manually adjust system settings.

7.7 Information Security

Ensuring the security of user data has been a primary consideration in the design of the OnBeat system. Several mechanisms have been implemented to safeguard sensitive information during both transmission and storage:

- **Password Encryption:** User passwords are never stored or transmitted in plain text. Instead, passwords are encrypted before being sent to the server and securely hashed for storage in the database. This ensures that even in the event of unauthorized database access, user credentials remain protected.
- **Secure Communication Channel:** All communication between the mobile application and the server is conducted over a secure WebSocket connection (wss). This protocol provides end-to-end encryption, ensuring that sensitive data such as authentication details, workout information, and music preferences cannot be intercepted or tampered with during transmission.

7.8 Risk Management

The OnBeat system integrates hardware, real-time algorithms, and mobile application logic, which naturally introduces several risks that can affect functionality and user experience. To ensure stability and reliability, the project team identified the main risks early in development, prioritized them, and implemented mitigation strategies as part of the ongoing testing and development cycle.

Main Risks and Mitigations

1. Hardware and Software Compatibility (High Priority)

- **Risk:** The Colmi R06 smart ring and the mobile application must operate reliably across different Android devices, operating system versions, and Bluetooth implementations. Compatibility issues could lead to unstable connections or limited accessibility.
- **Mitigation:** We have tested the application on multiple Android devices with different OS versions and Bluetooth chipsets to confirm stable operation and cross-device compatibility.

2. Data Accuracy and Real-Time Processing (High Priority)

- **Risk:** Since the system depends on heart rate (HR) data for adaptive music control, inaccuracies in sensor readings could reduce the effectiveness of training feedback. Real-time response is also critical for user safety and training optimization.
- **Mitigation:** We have tested the Colmi R06 ring against medically reliable devices with known accuracy (e.g., pulse oximeter) to validate HR measurement quality. The backend processing pipeline has been optimized to ensure real-time data handling without significant latency.

3. Environmental and Sensor Stability (Medium Priority)

- **Risk:** Environmental factors such as sweat, skin temperature, and rain can influence sensor reliability and HR accuracy.
- **Mitigation:** We have defined wide acceptable BPM zones in the algorithm to reduce false triggers and implemented visual indicators (color-coded HR feedback) to warn users regarding their HR readings.

4. Load and Scalability (Medium Priority)

- **Risk:** The system architecture is designed for multiple concurrent users, but testing capacity is currently limited by available hardware (only three smart rings available for evaluation).
- **Mitigation:** We have validated the WebSocket server with simulated traffic to confirm its ability to handle multiple concurrent sessions. While large-scale validation with many rings remains pending, the backend is already prepared for scalability.

Risk Management Plan

Our overall plan follows a **proactive test-and-validate approach**, supported by continuous risk monitoring and mitigation:

- **Early identification:** Risks were identified and prioritized during the Software Requirements Specification (SRS) phase.
- **Built-in mitigation:** Mitigation strategies were incorporated directly into development (e.g., real-time processing validation, cross-device tests).
- **Iterative validation:** Continuous testing ensures that both hardware and software risks are monitored throughout the project lifecycle.
- **Deferred risks:** Remaining risks (such as large-scale load testing) are documented for follow-up once additional hardware becomes available.

Since OnBeat operates in a non-critical, consumer-oriented domain, our risk management strategy balances practical assurance with development efficiency. By combining incremental validation with targeted mitigations, we ensured the system remained reliable and secure within its intended scope, without the overhead of formal proofs.

7.9 Exceptions Management

Exception management in OnBeat ensures that errors are handled gracefully without disrupting the workout experience. The system is designed to capture and respond to errors in three main areas: user authentication, Bluetooth connection with the smart ring, and backend server operations.

Login Exceptions

During authentication, the following exceptions are addressed:

- **Invalid credentials** - if the username or password are incorrect, the user is notified with a neutral error message without revealing sensitive details.
- **Network or server errors** - if the server is unavailable, the user is informed and may continue in offline mode until the connection is restored.

Ring Connection Exceptions

Because the system relies on real-time heart rate monitoring, exceptions in Bluetooth communication must be managed carefully:

- **Scanning** - errors may occur if Bluetooth permissions are missing, the adapter is disabled, or no devices are found. The system prompts the user with clear instructions (e.g., enable Bluetooth, move closer to the ring).
- **Connection** - exceptions such as connection timeout, bonding failure, or device incompatibility are detected. The system retries a limited number of times before requesting user action.
- **Disconnection** - if the ring disconnects unexpectedly, the app automatically attempts to reconnect. Workout data collected locally is buffered to avoid loss during reconnection attempts.

Server Exceptions

On the backend, exceptions are managed to preserve data integrity and system responsiveness:

- **Validation errors** - if client requests are malformed or incomplete, the server responds with structured error messages.
- **Database errors** - timeouts or conflicts are retried automatically; persistent failures trigger logging and alerts.

7.10 Versions Control & Code

Github is used to handle the system's files and repositories. You can find the link [here](#).

We added the demo of our system in action [here](#).

7.11 Project Management

To-Do				
☐	Task		Text	Timeline
☐	login and registration page	+	Nikolai	Mar 3 - 6
☐	safe login with hash	+	Lotem	Mar 6 - 9
☐	http requests	+	Nikolai	Mar 6 - 16
☐	fixes and bugs	+	Submission	Mar 20 - 27
☐	testing	+	Testing	Mar 20 - 27
☐	Beta submit for initial review	+	Submission	-
☐	Beta final submission	+	Submission	-
☐	implement speed	+	Lotem	Apr 7 - 11
☐	admin page - trainer	+	Nikolai	Apr 7 - 17
☐	logging music data	+	Maayan	Apr 7 - 22
☐	logging ring data	+	Lotem	Apr 7 - 20
☐	validation report submit for initial review	+	Submission	-
☐	testing and bug fixes	+	Testing	-
☐	CODE FREEZE	+		-
☐	validation report final submission	+	Submission	-

Figure 29: Gantt-1

To-Do					
☐	Task		Text	Timeline	Due date
☐	Build the user interface	⊕	Nikolai	Jan 9 - 16	Jan 16
☐	Integrate basic communication between SmartRing with the app	⊕	Lotem	Jan 9 - 16	Jan 16
☐	Categorize songs by tempo and map songs to main genres	⊕	Maayan	Jan 1 - 5	Jan 12
☐	SDD doc - submit for initial review	⊕		-	Jan 23
☐	Plan algorithm for real-time BPM based music adjustment	⊕	Maayan	Jan 27 - Feb 9	Feb 18
☐	Implement full bluetooth communication between smartring and app	⊕	Lotem	Jan 27 - Feb 16	Feb 18
☐	set up API calls to fetch real time metrics from the smart ring	⊕	Nikolai	Jan 27 - Feb 7	Feb 18
☐	set up a database for storing user performance data	⊕	Lotem	Jan 20 - 22	Feb 28
☐	implement algorithms for generating performance summaries	⊕	Maayan	Feb 10 - 18	Feb 28
☐	verify accuracy of smartring pulse	⊕	Lotem	Feb 13 - 27	Feb 28
☐	sync performance data to the backendserver for analysis and logs	⊕	Nikolai	Feb 10 - 26	Feb 28
☐	alpha submit for initial review	⊕	Submission	-	Mar 27
☐	alpha final submission	⊕	Submission	-	Mar 1
☐	Adding better UI	⊕	Nikolai	Mar 3 - 11	Mar 9
☐	presenting the steps and distance data from the smartring	⊕	Lotem	Mar 3 - 15	Mar 14
☐	implementing new model	⊕	Maayan	Mar 3 - 19	Mar 20
☐	extending the algorithm to handle cold start - similar users	⊕	Maayan	Mar 3 - 14	Mar 20

Figure 30: Gantt-2

8 SYSTEM VALIDATION

The validation process of the OnBeat system represents a critical stage in ensuring the reliability, accuracy, and usability of its core functionalities. It serves not only to confirm that each component operates according to its specifications, but also to assess how effectively these components integrate to deliver a seamless, personalized workout experience. By methodically evaluating system performance under realistic usage conditions, the validation process offers valuable insights into the system's strengths, areas for improvement, and readiness for deployment.

Validation Overview

OnBeat's validation activities focused on verifying the functional correctness, responsiveness, and robustness of its interconnected components. This included empirical measurements of sensor accuracy, assessments of algorithmic adaptability and evaluation of user interface accessibility. Testing was conducted in conditions that replicated real-world usage scenarios, ensuring that the system's behavior accurately reflected its intended use.

Validation Methodology

The validation process was structured into targeted test categories, each addressing a critical aspect of the system's operation:

- **Heart Rate Accuracy Validation** - comparing the wearable ring's measurements with a clinically validated reference device.
- **Step Count, Distance, and Speed Estimation** - assessing the accuracy of motion data against ground-truth measurements.
- **Functionality Tests** - verifying the functional operation of the system and its components and internal communications.
- **Data Stability** - monitoring persistence, synchronization, and error handling for continuous operation.
- **Load Testing Limitations** - determining system behavior under increased simultaneous user sessions and data throughput.
- **Algorithm Correctness** - validating the music selection and tempo adaptation logic under varied workout profiles.

An iterative validation approach was employed, with testing phases integrated throughout the development lifecycle. This allowed for early detection of defects, rapid incorporation of feedback, and continuous refinement of the system to align with design objectives and user expectations.

Heart Rate Accuracy Validation

The heart rate monitoring capability of OnBeat was validated using the COLMI R06 smart ring, which integrates the VCare VC30F photoplethysmography (PPG) sensor. This sensor captures continuous biometric readings at a sampling rate of 1 Hz, enabling real-time adaptation of music tempo to the runner's physiological state. Validation testing was conducted through side-by-side comparisons with a medical-grade BioLight M70 fingertip pulse oximeter (± 1 BPM accuracy) under both resting and active exercise conditions.



Figure 31: BioLight M70 pulse oximeter and Heart Rate Measurements from the Ring

Heart rate readings from the ring and reference device were recorded simultaneously, and deviations were calculated for each trial.

Test Results

Test	Pulse Oximeter (BPM)	OnBeat (BPM)	Δ (Difference)
T1	76	74	-2
T2	75	73	-2
T3	109	106	-3
T4	105	108	+3
T5	108	106	-2
T6	110	115	+5

Figure 32: Table of heart rate trial results.

Average difference: **-0.17 BPM**

Mean Absolute Error (MAE): **2.83 BPM**

Results indicated that the VC30F maintained a mean absolute error of 2.83 BPM, with a maximum observed deviation of ± 5 BPM. Given the acceptable accuracy threshold for fitness applications, this performance is sufficient for driving real-time tempo adjustments. Bluetooth Low Energy (BLE 5.0) connectivity was stable throughout all tests, and no perceptible latency. These results confirm that the heart rate monitoring subsystem is both accurate and responsive for its intended purpose in OnBeat's adaptive music feedback loop.

Step Count, Distance & Speed Estimation Validation

OnBeat's step count and pace estimation capabilities are provided by the STK8321 3-axis accelerometer embedded in the COLMI R06 ring. The device firmware processes motion patterns to estimate step count, distance, and activity state, transmitting this data to the OnBeat mobile application via BLE. Although not highly precise, the step and distance data from the STK8321 sensor in the Colmi R06 ring offers sufficient fidelity for OnBeat's real-time decision-making. It supports the system's goal of enhancing the runner's experience by syncing music tempo to both heart rate and physical activity without requiring GPS or external tracking tools.

Functionality Testing

Functionality testing was carried out to ensure that all system components operated correctly both individually and when integrated. The goal was to verify that the OnBeat application and its backend fulfilled the functional requirements defined earlier, while maintaining stable internal communication between the mobile application, backend services, wearable device, and database. The testing process was divided into three complementary stages: **unit testing** for isolated component verification, **integration testing** for inter-module communication and workflow validation, and **GUI testing** for user interface correctness and usability.

1. Unit Tests

Unit testing focused on validating the **correctness of discrete components** within the OnBeat architecture, such as the workout selection, BPM data parsing module, Bluetooth communication handler, and local data storage functions. Each component was tested against predefined input-output pairs or operational criteria, ensuring its behavior matched the expected results in isolation from other parts of the system. Test cases covered both **nominal scenarios** and **edge cases**, with results systematically recorded for traceability. The collected data demonstrated that all tested units performed according to specification, confirming the stability of the underlying application logic.

2. Integration Tests

Integration testing evaluated the **interaction between** the mobile **application**, **backend server**, machine learning module, **database**, and Bluetooth-enabled smart ring. Scenario-driven test cases were developed to mimic realistic user workflows, including initiating a run, streaming heart rate data, dynamically adjusting music tempo, and saving performance metrics to the database. The tests verified the correctness of data transmission, system state transitions, and error handling across component boundaries. Particular emphasis was placed on confirming that **inter-module communication** remained consistent even under varying network conditions. The recorded results showed successful coordination between all major system elements, with no observed data loss or state mismatches.

The integration validation confirmed that OnBeat's core components - mobile app, server, database, wearable device interface, and adaptive music logic - interoperated seamlessly, delivering the intended end-to-end functionality under expected usage conditions.

3. GUI Tests

Graphical User Interface (GUI) testing was performed to ensure that all user-facing elements met their functional and usability requirements. Using an exploratory testing approach supplemented with a checklist, testers systematically interacted with critical **UI components**, including navigation menus, configuration forms, real-time metric displays, and playback controls. The evaluation assessed **correct behavior** and **responsiveness** and **layout consistency**, error-free interaction, and adherence to design specifications.

The GUI testing results confirmed that all tested interface elements operated correctly, displayed accurate information, and responded predictably to user actions. The application's interface was judged to be intuitive, responsive, and aligned with the defined usability requirements.

Conclusion: The functionality testing process confirmed that the OnBeat system fulfills its **core operational requirements**, both at the **component** and **integrated levels**. Unit tests demonstrated stable and predictable behavior in isolated modules, while integration testing validated reliable communication and data flow between the mobile app, backend server, database, wearable device, and adaptive music algorithm. GUI testing confirmed that the user interface is both functionally correct and aligned with the intended usability design. Collectively, these results indicate that the system delivers its intended features effectively, operates reliably under expected conditions, and provides a smooth user experience across its core functionalities.

Data Stability

The data stability validation assessed the OnBeat system's ability to maintain consistent operation, accurate data synchronization, and effective error handling during prolonged usage. Given that OnBeat adapts training intensity in real time based on biometric and movement data, it is essential that the data pipeline - spanning sensor acquisition, transmission, processing, and storage - remains stable under varied conditions.

Testing scenarios included both controlled indoor runs and outdoor sessions in variable environmental conditions such as heat, cold, and high humidity. As anticipated, certain factors were found to influence measurement reliability: for instance, sweat or cold skin occasionally reduced the accuracy of the photoplethysmography (PPG) heart rate readings, while heavy rain introduced minor fluctuations due to interference with capacitive and optical sensors, despite the Colmi R06 ring's water resistance.

To mitigate these effects, the system implements a broad acceptable BPM tolerance range before triggering adaptive responses and provides real-time visual HR indicators that alert the user when sensor readings are potentially unreliable. Overall, the OnBeat system demonstrated strong data stability under most conditions, with environmental impacts well-managed through built-in tolerance thresholds and user feedback mechanisms.

Load Testing Limitations

The load testing phase aimed to evaluate the OnBeat system's ability to sustain functional performance under conditions of increased concurrent usage. The backend infrastructure, built on WebSocket-based communication and session management, was designed to support multiple simultaneous users in alignment with the non-functional requirements specified in the SRS.

Due to hardware limitations - specifically, access to only three Colmi R06 smart rings - load testing was constrained to a maximum of three concurrent sessions during validation. Within this scope, the system maintained stable connections, processed biometric and motion data in real time, and adapted music playback without noticeable latency or degradation in performance.

While the results indicate that OnBeat is technically prepared for multi-user operation, full-scale stress testing remains incomplete. Comprehensive load validation at higher concurrency levels will be necessary in future development phases once additional hardware becomes available.

Algorithm Correctness

The algorithm correctness validation aimed to confirm that the adaptive music selection and tempo adjustment logic within the OnBeat system behaves according to its design under realistic workout conditions. The algorithm's primary role is to respond to biometric input—specifically heart rate (BPM), workout stage, and running speed—by selecting or adjusting music tracks in real time. Correctness in this context means that for each detected deviation (e.g., a BPM drop),

the system consistently triggers the appropriate musical adaptation (song switch or tempo change), thereby supporting runner motivation and stabilizing performance.

Validation Procedure

Validation was conducted through live running sessions with 20 participants, each completing two comparable runs: one without music and one with music. This within-subject design ensured that each participant served as their own control, making it possible to directly compare performance and heart rate stability across conditions.

To capture validation data, the admin dashboard recorded both physiological and system-level metrics:

- **Physiological performance:** start/end BPM, step count, estimated distance.
- **Algorithm responses:** timing and triggers for run_score changes, including reasons for each score event.
- **Music adaptation:** song playback history with entry/exit reasons, effectiveness classification, and tempo adjustments.

In addition, the system logged real-time events such as score updates (every few seconds) and music adaptation decisions when deviations from the target HR zone were detected. This procedure allowed us to validate the algorithm's performance under varied participants, fitness levels, and music preferences.

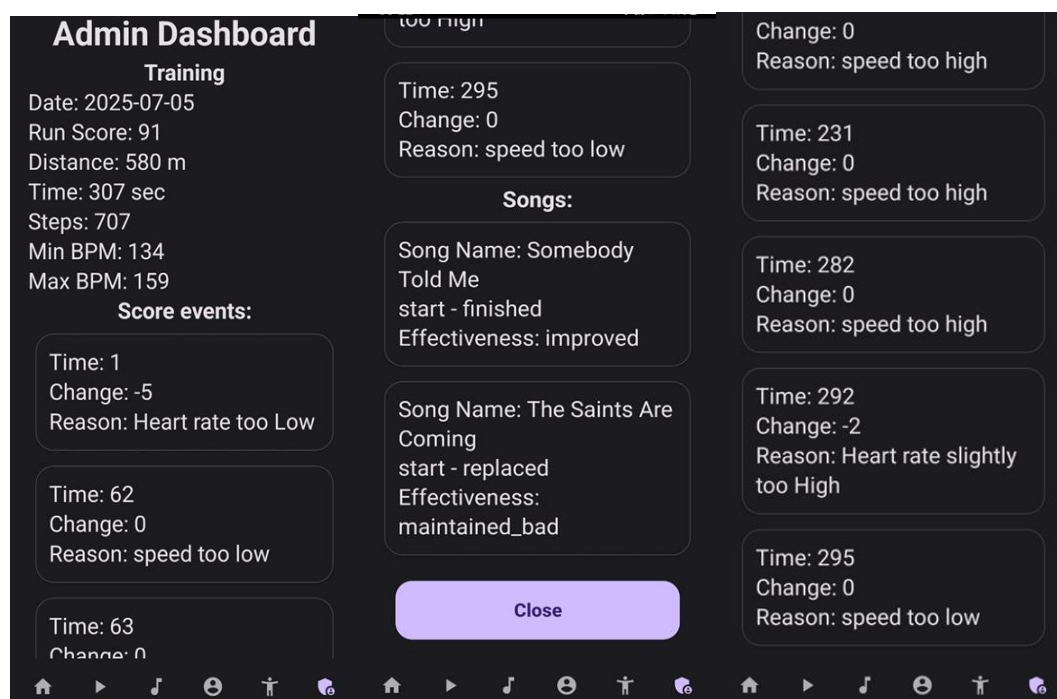


Figure 33: Admin dashboard displaying run statistics and scoring events.

Figure 34: Song history with trigger reasons and effectiveness classification.

Figure 35: Run events log - Small BPM deviation resulting in a 2-point deduction.

Observed Behavior and Results

The experimental runs demonstrated clear differences between conditions. Across the 20 participants:

- Run Score (primary metric):
 - Without music: mean = 77.3
 - With OnBeat: mean = 87.95
 - Average improvement = +10.65 points (+13.8%)
 - 17 of 20 participants (85%) improved their run_score with OnBeat.
- Zone Compliance (derived measure):
 - Without music: 81.4% in the HR target zone
 - With OnBeat: 92.6% in the HR target zone
 - Improvement = +11.2% compliance.
- Distribution of improvements: The majority of participants showed positive changes, with only a small minority showing little or no effect.

Average BPM Deviations per Phase

- **Begin**
 - Without Music: 0.95
 - With Music: 0.15
- **Middle**
 - Without Music: 4.90
 - With Music: 2.00
- **End**
 - Without Music: 2.70
 - With Music: 1.25

Differences (With – Without):

- Begin: -0.80
- Middle: -2.90
- End: -1.45

As shown above, the average number of deviations decreased in all phases when music was played. The most pronounced effect was observed in the middle phase (-2.9 deviations on average), when fatigue typically causes instability. The begin and end phases also showed consistent improvements (-0.8 and -1.45 respectively). This indicates that OnBeat algorithm effectively reduced BPM drops throughout the workout by adapting music in real time.

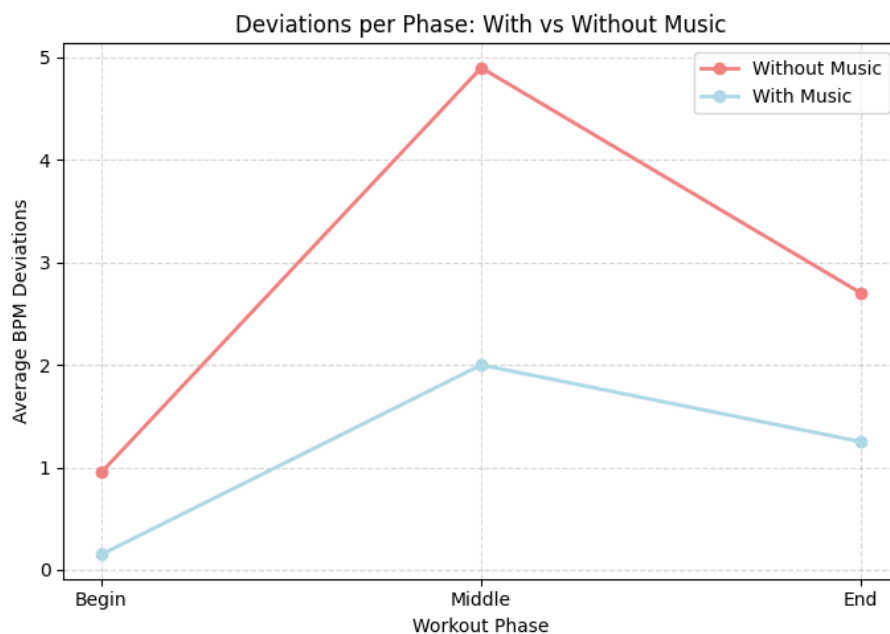


Figure 36: Diviations per phase with vs without music

Correlation Potential

Beyond validating correctness, the data opens potential avenues for correlation analysis:

- **Music features and effectiveness:** Each run stored a detailed Session_Songs log (genre, tempo, entry/exit reasons, effectiveness tags). This enables correlation between song characteristics (e.g., fast tempo rock vs. slower pop) and improvement in run_score.
- **User factors:** Since participants varied in fitness level, running speed, and personal music preference, future analyses could test whether certain runner profiles respond more strongly to tempo adaptation.
- **Temporal dynamics:** The logs allow segmentation of early, middle, and late run phases, making it possible to study when OnBeat interventions are most effective.
- **Cross-metric validation:** Linking run_score improvements to other measures (distance covered, step count, or pace stability) could provide a more holistic understanding of how music adaptations translate into performance gains.

These correlations were not the primary focus of this validation, but the recorded data provides a strong basis for further exploration.

Conclusion

The validation confirmed that the OnBeat system achieves its core objective: adaptive music playback reduces BPM drops and improves heart rate stability during running. Using a within-subject design (20 participants, each completing runs with and without OnBeat), we observed:

- Significant average improvement in run_score (+13.8%)
- Increased HR zone compliance (+11.2%)
- Clear majority of participants benefiting (85% improved)

These findings support the correctness of the algorithm and demonstrate its practical value under real-world conditions. Moreover, the comprehensive logging framework not only validated the system but also established a foundation for richer correlation analyses in future work.

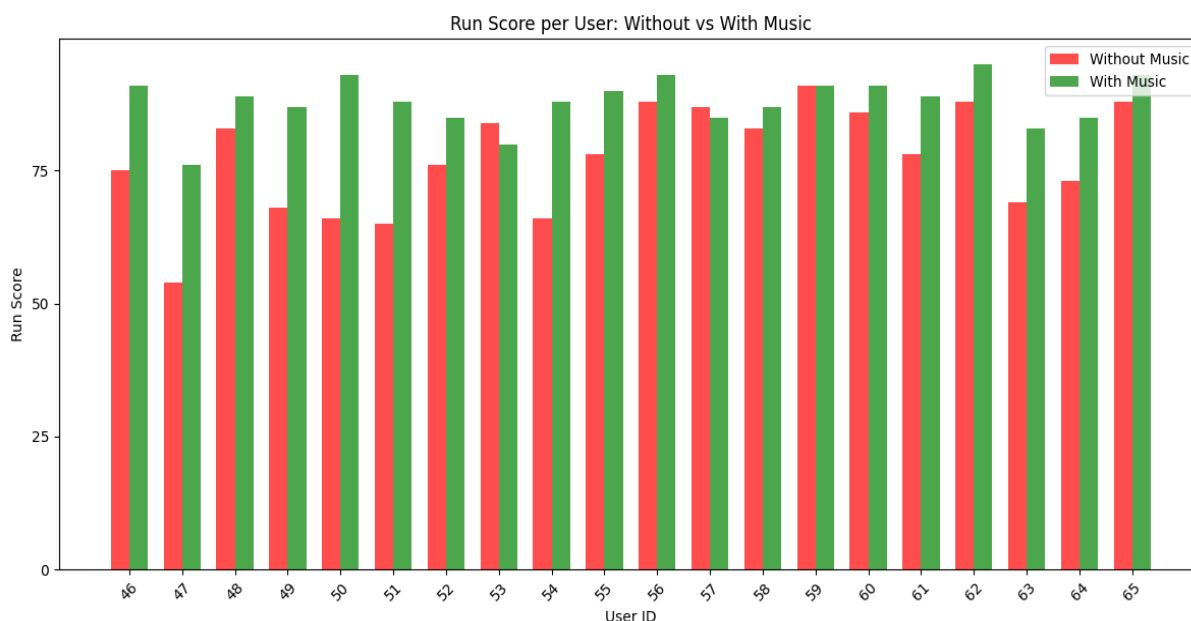


Figure 37: Run score per user with and without music.

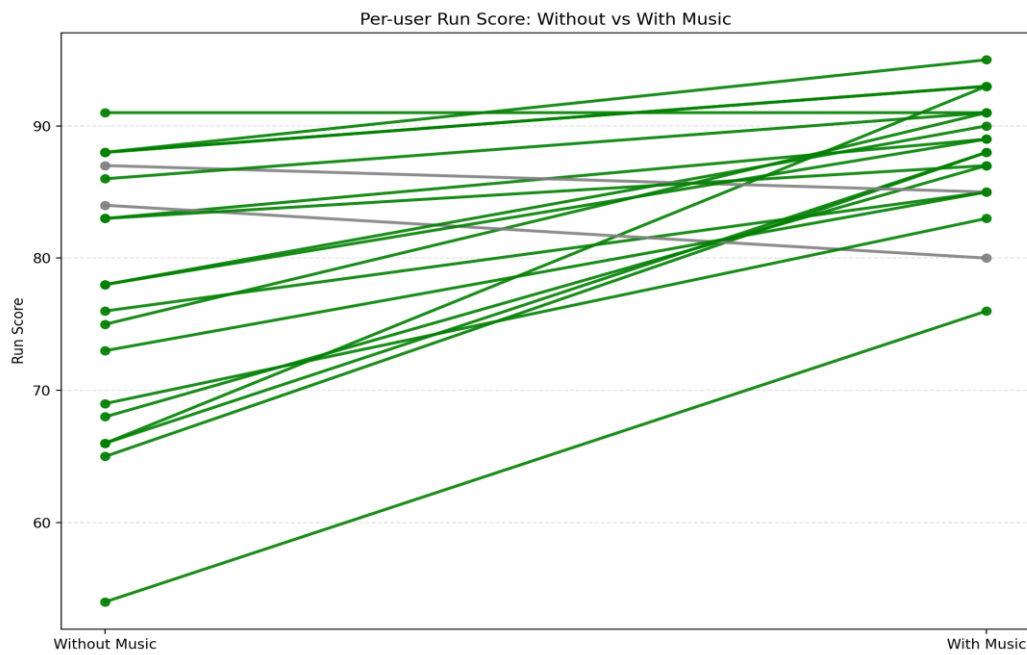


Figure 38: Run score changes per user between without and with music.

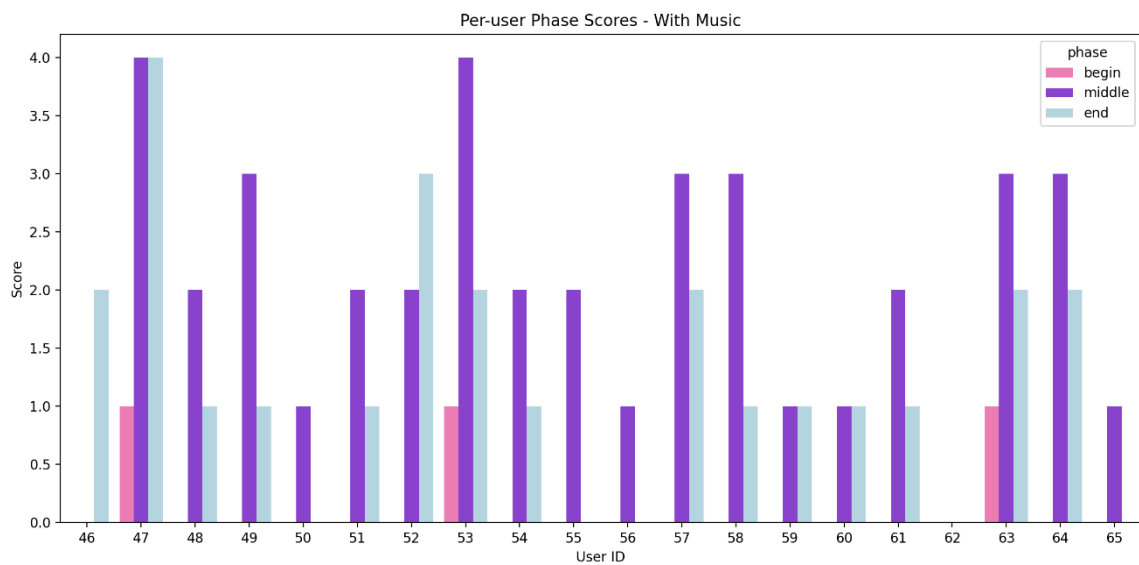


Figure 39: Deviation in BPM for every phase of users runs with music.

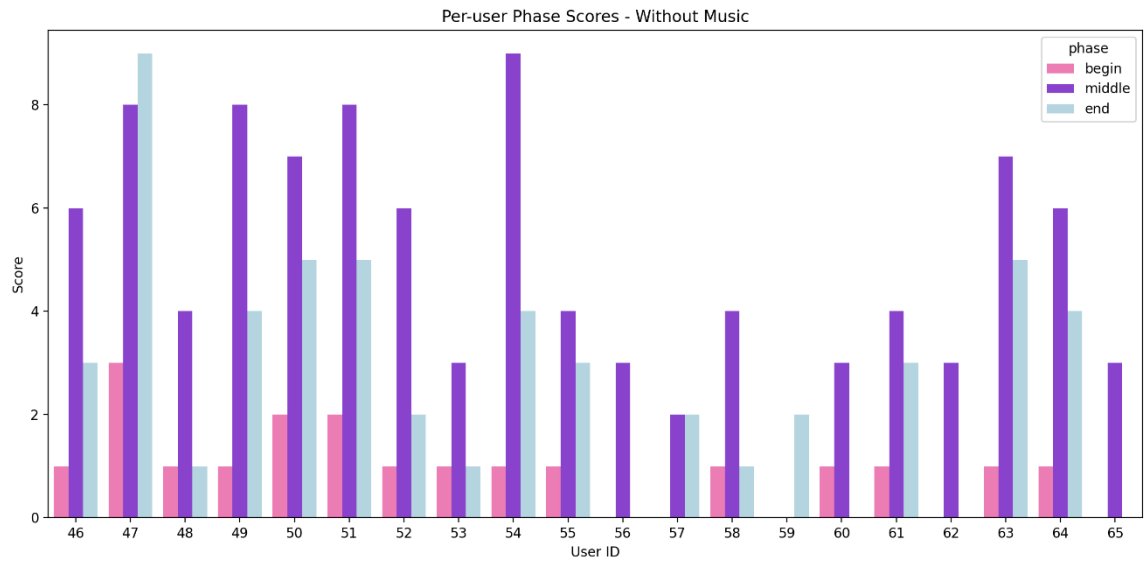


Figure 40: Deviation in BPM for every phase of users runs without music.

9 SUMMARY, EVALUATION, CONCLUSIONS AND FUTURE WORK

9.1 Summary

The OnBeat system was developed to enhance workout experiences through real-time heart rate monitoring and adaptive music playback. Using the Colmi R06 smart ring as the primary biometric device, the system collects heart rate data via Bluetooth and transmits it to the OnBeat mobile application. This data is then processed by a Python-based backend server, which analyzes the user's BPM and selects appropriate songs from a preloaded music database. A LightGBM-based algorithm was employed to improve the matching between user heart rate, song tempo, and workout stage by leveraging prior workout history and user similarity. The selected tracks are delivered to the app, where playback speed and transitions are adjusted to match the user's workout intensity. The system also stores workout histories, song performance statistics, and training session data to provide valuable insights for both runners and trainers.

9.2 Evaluation and conclusions

- **Sensor Accuracy:** Heart rate readings from the Colmi R06 ring were compared against a medical-grade reference device. Results showed an acceptable deviation within fitness standards.
- **Algorithm Performance:** The LightGBM algorithm successfully modeled the relationship between workout data, heart rate trends, and song effectiveness. It improved song selection by identifying which tracks helped users stay within target BPM ranges and by adjusting playback during deviations.
- **User Experience:** The mobile application provided seamless playback control, real-time responsiveness, and session tracking. Logs of song choices and outcomes were successfully recorded for later analysis.
- **System Reliability:** Testing confirmed stable Bluetooth connectivity, effective server communication, and consistent storage/retrieval of workout and music data.

The OnBeat project demonstrates that adaptive music playback can positively influence workout performance by aligning musical rhythm with physiological signals. By combining biometric sensing, LightGBM-driven song selection, and real-time playback control, the system provides a dynamic, motivating, and personalized workout experience. The successful integration of wearable technology, mobile application, and backend processing highlights the system's potential as a scalable platform for future fitness innovations.

9.3 Future work

Several opportunities exist to expand and refine the OnBeat system:

- **Multi-Device Support:** Extend compatibility to additional biometric wearables beyond the Colmi R06.
- **Cross-Platform Expansion:** Add support for IOS devices to broaden the system's accessibility.
- **Expanded Music Integration:** Incorporate online streaming services alongside the local music database for a wider song selection.
- **Personalized Recommendations:** Enhance the LightGBM model or combine it with deep learning to better adapt to user preferences and historical performance.
- **Trainer Dashboard:** Develop advanced visualization tools for trainers to monitor multiple runners in real-time.

- **Gamification Features:** Introduce challenges, progress badges, and community features to increase motivation and engagement.
- **User Interface Redesign:** Improve the mobile app UI for a more intuitive and visually engaging experience, tailored for both runners and trainers.
- **Energy Efficiency:** Optimize Bluetooth communication and data processing to reduce device battery consumption during long workouts.

10 REFERENCES

- [1] Karageorghis, Costas I. "The scientific application of music in exercise and sport." *Sport and exercise psychology* (2015): 1-6.
- [2] Thakare, Avinash E., Ranjeeta Mehrotra, and Ayushi Singh. "Effect of music tempo on exercise performance and heart rate among young adults." *International journal of physiology, pathophysiology and pharmacology* 9, no. 2 (2017): 35.
- [3] Cassidy, G. G., and Raymond AR MacDonald. "The effects of music on time perception and performance of a driving game." *Scandinavian journal of psychology* 51, no. 6 (2010): 455-464.
- [4] Maddigan, Meaghan E., Kathleen M. Sullivan, Israel Halperin, Fabien A. Basset, and David G. Behm. "High tempo music prolongs high intensity exercise." *PeerJ* 6 (2019): e6164.
- [5] Karageorghis, Costas I., Leighton Jones, David-Lee Priest, Rose I. Akers, Adam Clarke, Jennifer M. Perry, Benjamin T. Reddick, Daniel T. Bishop, and Harry BT Lim. "Revisiting the relationship between exercise heart rate and music tempo preference." *Research quarterly for exercise and sport* 82, no. 2 (2011): 274-284.
- [6] Karageorghis, Costas I., Leighton Jones, and Daniel C. Low. "Relationship between exercise heart rate and music tempo preference." *Research quarterly for exercise and sport* 77, no. 2 (2006): 240-250.

תקציר

מערכת OnBeat פותחה כדי לשלב ניטור דופק בזמן אמת באמצעות טבעת חכמה מסוג 06Colmi R עם אפליקציית מובייל. המערכת אוספת נתוני דופק מהמשתמש ומשדרת אותם לשרת לעיבוד. על סמך ניתוח הקצב, נבחרות יצירות מוזיקליות מקומיות ומותאמות כך שהקצב המוזיקלי יתאים לדופק המשתמש. בדרך זו נשמרת חווית אימון אישית, מעודדת וממוקדת מטרה.

בנוסף, המערכת שומרת נתוני ריצה וסטטיסטיקות על ביצועי המשתמש, ומספקת כלים לניתוח לטווח ארוך ולשיפור תכניות האימון האישיות.

הכרת תודה

ברצוננו להודות לד"ר מרסלו שיכמן ולד"ר דביר רוס על הליווי, ההדרכה והתמיכה לאורך הפרויקט. לבסוף, נרצה להודות לראש המחלקה, פרופ' אביבית לוי, על התמיכה וההכוונה לאורך תקופת הלימודים.



OnBeat

הנדסת תוכנה

B.Sc. ספר פרויקט גמר לתואר ראשון בוגר במדעים

מאת: מעיין בביוף, ניקולאי מלניצ'ב, לוטם יעקוביאן
מנחה: דר' מרסלו שיכמן, דר' דביר רוס

18/08/2025