University of Waterloo
Faculty of Engineering
Department of Electrical and Computer Engineering

TempoTracks

Group 2023.017

Garrick Clarke	g3clarke@uwaterloo.ca	20822323
Rizwan Kassamali	rkassama@uwaterloo.ca	20834079
Alexander Ma	a37ma@uwaterloo.ca	20822811
Matthew Panizza	mpanizza@uwaterloo.ca	20822324
Kuba Rogut	krogut@uwaterloo.ca	20844610
Sterlyn Tang	s224tang@uwaterloo.ca	20859326

Consultant: Douglas Wilhelm Harder

07 June 2023

Table of Contents

1 High Level Description 3

1.1 Motivation 3

1.2 Project Objective 3

1.3 Block Diagram 4

1.3.1 iOS Mobile App Subsystem 4

1.3.2 WatchOS Application Subsystem 5

1.3.3 Backend Service Subsystem 5

1.3.4 Music Service Subsystem 5

2 Project Specifications 5

2.1 Functional Specifications 6

2.2 Non-Functional Specifications 7

3 Risk Assessment 8

4 References 10

1 High Level Description

1.1 Motivation

There is no easy way to create a playlist that would match your workout pace, whether that is to keep up with a target pace to reach a goal or for the music to keep up with the runner's pace. Furthermore, it is even more of a challenge, if the pace of the workout varies, songs will not speed up and slow down to keep the runner going at the current pace. Music can provide ergogenic, psychological, and psychophysical benefits during physical activity, especially when movements are performed synchronously with music [1]. The goal of this application is to help athletes in their workouts keep a specified tempo or be able to reach a time goal where they can run to a modular soundtrack. A study showed that music speed is directly related to how athletes perform. Speeding up the music program increased distance covered/unit time, power, and pedal cadence from 0.7-3.5% respectively; slowing the program produced falls of 3.8-9.8% [2]. By harnessing the power of music, we are able to help individuals train better and become stronger.

1.2 Project Objective

The goal of this project is to give the user an application where they will be able to have music the adapts to the pace of the workout or that pushes the user to hit a pace goal. An iPhone application and an Apple Watch application will be created, and these apps will work together to provide support for the user. Athletes will not have to carry their phones while working out and can just go with their watch to the training session. The phone application will be used to create custom workouts, import the music library, view and track progress over a period of time and more. All workouts will be saved and will be able to synchronize with Apple Health and merge with the user's current workout history.

The Apple ecosystem was chosen for this project due to a large user base, a vast selection of music via Apple Music and the ability to access and modify music data by using Apple Music Toolkit. The project will leverage the fact that users are still using media they are familiar with, as they are able to use their own music playlists. A user can create a running or triathlon plan with ease and let the app do the rest. It will process these playlists by sorting them by beats per minute and pitch, according to the user's cardio plan. In most cases, the beats per minute will peak near the middle of their run and decrease exponentially at the end. If the user wants to do a different style of cardio, the app will accommodate that as well.

1.3 Block Diagram

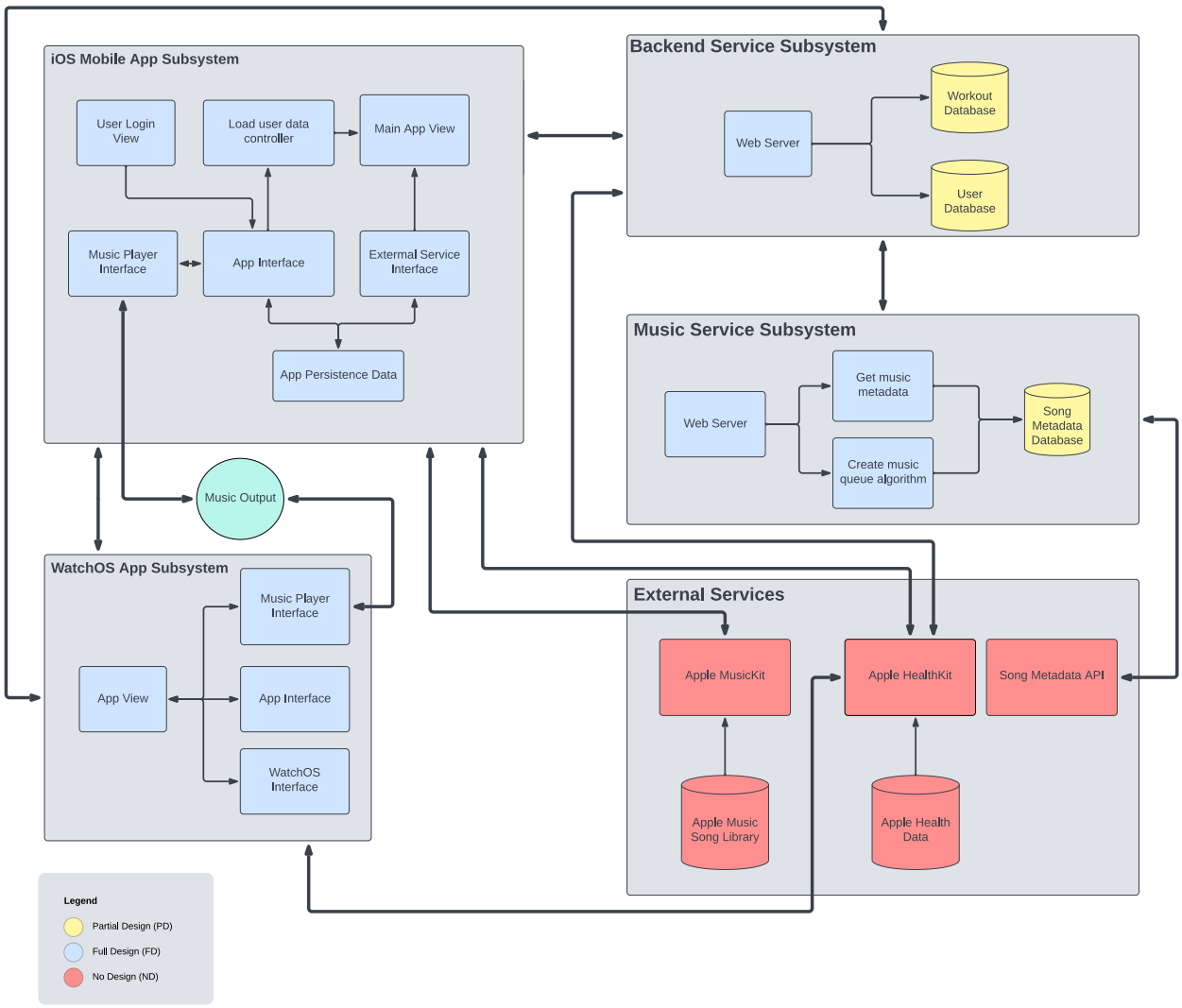

Figure 1: *Block Diagram of TempoTracks system*

1.3.1 iOS Mobile App Subsystem

The iOS mobile application will be the main way a user interacts with our application. From this interface the user will be able to create, start and stop their workouts, which will also create the queue of music for their workout. To access the user's health data, the application will prompt the user to give us access to their stored health data using Apple's HealthKit. This app will be connected to the user's Apple Music, where we will be controlling the play of music along with the playback speed of the music. The app will also have to connect to the user's Apple watch, so that the data can sync across the devices. Also, we must leverage the iPhone's local data storage so that parts of our app can work offline.

1.3.2 WatchOS Application Subsystem

The watch application is the other way the user will interact with our application. This application will act as the supplementary application to the main iOS application and will only contain the essential features for a user to perform their workouts. It will be able to create, start and stop workouts, along with playing the music queue. It should also be connected to the user's health data. Furthermore, when separated from the paired iPhone, the watch should be able to save any workout data and resync when the connection is re-established.

1.3.3 Backend Service Subsystem

The backend service subsystem will be responsible for connecting the database to a webserver where the front-end applications will be able to request information from the backend. The backend service will consist of two databases. The first database will house the user data, including login information and the health data they choose to share with the application. The second database will have all the workouts for each user and the associated data for each workout. The webserver will use these databases and create both a public and private interface that will be used the iOS application and the WatchOS subsystem.

Furthermore, the backend service subsystem will connect to the music service subsystem to get the songs and playlist for each workout as well as the data and metrics of each song such as beats per minute. Lastly, the backend system will need the information from Apple HealthKit. Health data such has the heart rate, and the stride length of the user will be used in real time to adjust and tune the tempo of the music.

1.3.4 Music Service Subsystem

The last subsystem is the music service subsystem. This subsystem will provide the music from the backend system and this subsystem is responsible for obtaining all the metadata by using the external API. Furthermore, this subsystem is responsible for speeding up and slowing down the music accordingly. This subsystem will ensure that the output audio still is pleasant to listen to by adjusting other factors such as pitch.

2 Project Specifications

In this section the functional and non-functional requirements for TempoTracks will be discussed. Table 2-1 details the function requirements that are needed and Table 2-2 outlines the non-functional requirements. Each specification will be classified as essential or non-essential.

2.1 Functional Specifications

Specification	Description	Classification
Access to health data	The user's health data such as heart rate and stride length will be required and imported from Apple Health to be able to create a playlist matched to their intensity.	Essential
Variety of music	A large variety of music will be needed as the tempo in a workout can change based on the type of workout and the phase of the workout such as warm up, high intensity or cooldown.	Non-Essential
Import from Apple Music	Apple Music is required as the user must import their music library from Apple Music. This import allows us to control the tempo and pitch of the tracks through Apple Music Kit. The music will still be played though the Apple Music application.	Essential
Ability to speed up/slow down music tempo	In order to match the pace and tempo of the user, the application must be able to adjust the pitch and tempo of the music. This can be achieved using Apple Music Kit where parameters such as pitch and tempo can be set for the songs in the workout playlist.	Essential
Evaluate desired speed to determine next song choices	A graph will be used to determine the desired speed at any given point in the run. This function should choose an appropriate song for the run and modify the speed and the pitch as necessary.	Essential
Create workout for user	The user must be able to create a workout within the app. The workout should contain the activity, the duration, the intended goals, and details specific to the activity. The users must also pair songs or a playlist to this workout through the Apple Music import.	Essential
Start workout	The application must immediately start the workout upon the user's command and start to play the music with the correctly adjusted tempo.	Essential
Play song to match user's pace	The application must be able to play a song to match the user's pace.	Essential
Present workout and relevant health data to user	The application must show the workout data to the user on the watch in real time. The app must also show the complete workout data and the health data such as heart rate variability on the phone app after the workout is complete.	Non-Essential

Table 2-1: *Functional Specifications*

2.2 Non-Functional Specifications

Specification	Description	Classification
Watch Application Size	The watch app should not be more than 300MB as smart watches tend to have less storage space than other devices such as smart phones.	Essential
Phone Application Size	The phone app should not be more than 700MB since larger app sizes can defer users from downloading the app.	Essential
Battery Consumption	The watch app should not consume more than 15% of battery capacity per hour. Watches tend to have a smaller battery size in comparison to other devices (e.g smart phone). Since the watch screen will be on during the workout, this will also place extra strain on the battery.	Essential
Internet Access	Both the mobile and watch app should utilize the internet to allow the user to login, sync their Apple Music library and fully utilize some of the other services/subsystems.	Essential
Offline Usability	App could be accessed offline. This would be convenient for user but can be avoided with a data plan.	Non-Essential
Scalability	Must be able to support over 5000 users at the same time.	Non-Essential
Backend Server Uptime	The backend sever should be up and running 99% of the time to ensure that users are able to store and retrieve data which is not stored locally on their device.	Essential
Supports various Apple watches	The app should work on the Apple Watch series, the Apple Watch SE series, and the Apple Watch Ultra. The watch application must consider the difference in the 3 series of watches, specifically the always on display, screen size and their respective battery sizes.	Non-Essential
WatchOS / iOS supportability	The app should work on WatchOS 9.0.0 and future updates to the operating system. The phone app should work on iOS 16.0.0+.	Essential
Data Loss	If the health or accelerometer data is not updated, the app should continue playing music at the predetermined intervals set in the workout or at the last known pace for an open-ended workout.	Essential

Table 2-2: *Non-Functional Specifications*

3 Risk Assessment

The major risks associated with this project are highlighted below in Table 3-1.

Potential Risk	Description	Probability	Impact
Technical Risks	**Situation** - Application is too heavy for the Apple Watch to handle and iPhone companion app is needed. Not all group members have access to a MacOS system/laptop which is a requirement for building and developing on iOS and WatchOS.	High	Low
	Remediation – Split up group work based on strengths and device availability. Group members without MacBooks will be in charge of backend services and database management. Members with MacBooks will be responsible for creating the applications for the Apple Watch and iPhone.		
Group Member Availability	**Situation** – As the group consists of 4th year students taking a wide variety of challenging technical electives, it can be hard to coordinate times to meet up and collaborate on the project.	Medium	Medium
	Remediation – To solve this issue meeting times can be planned well in advance and have set agendas for each meeting which would reduce wasted time in meetings. Meetings can also be held online, and features-focused subgroups can be formed within the overall group, to better accommodate the availability of group members.		
Songs may not sound good at all speeds	**Situation** - Audio may not sound good if the user wants a really fast or slow pace for their workout. If the user has very slow songs in the playlist for a selected workout, we need to ensure that we reach the beats per minute (BPM) goal while ensuring that the music is still pleasant to listen to.	High	High
	Remediation – To fix this issue we predetermine the speed and beats per minute (BPM) of each song and align the tempo with the various intensity levels of the workout to minimize the impact on the audio. Furthermore, we can manipulate the pitch of the music to minimize the distortion in speed.		

Integration with heart rate	**Situation** – If the user has certain heart rate goals during a workout to be able to push themselves in a workout, we need the music to be able to match the tempo. The challenge here is to be able to read the heart rate live from the watch and use that to fine tune the tempo of the music in real time.	Medium	Medium
	Remediation – We can solve this by using heart rate zones where we only change the music's tempo if the user enters a higher or lower intensity zone.		
Music Licencing	**Situation** – Some audio tracks might require a Digital Rights Management and as such pitch scaling would not be available for those tracks.	High	Low
	Remediation - For these audio tracks, only the tempo will be adjusted accordingly. These tracks would also be used closer to the original tempo of the song as we cannot modify the speed of the song in a large capacity before the song is no longer pleasant to listen to.		

Table 3-1: *Risk Assessment*

4 References

[1] P.C Terry, C.I Karageorghis, A.M Saha, S. D'Auria, "Effects of synchronous music on treadmill running among elite triathletes", *Journal of Science and Medicine in Sport*, 2012. [Online] Available: https://www.sciencedirect.com/science/article/pii/S1440244011001186 (accessed Jun. 5, 2023).

[2] J. Waterhouse, P. Hudson, and B. Edwards, "Effects of music tempo upon submaximal cycling performance," *Wiley Online Library*, 2010. [Online] Available: https://onlinelibrary.wiley.com/doi/10.1111/j.1600-0838.2009.00948.x (accessed Jun. 7).