Автономная некоммерческая организация высшего образования «Университет Иннополис»

Autonomous noncommercial organization of higher education «Innopolis University»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА ПО НАПРАВЛЕНИЮ ПОДГОТОВКИ 09.03.01 – «ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

GRADUATE THESIS MAJOR: «COMPUTER SCIENCE»

Тема **Подбор музыки на основе показателей активности пользователя**

Topic **Training performance aware music recommendation**

Работу выполнил / Thesis is executed by

Бочкарев Григорий Игоревич
Bochkarev Grigory Igorevich

подпись / signature

Научный руководитель / Thesis supervisor

Цюй Цян
Qiang Qu

подпись / signature

Иннополис, Innopolis, 2017

Chapter 1. Introduction

Project background and history

For some athletes and for many people who engaged in sport, the music has an important impact on their performance. In the last 10 years, the body of research on workout music has swelled considerably, helping psychologists refine their ideas about why exercise and music are such an effective pairing for so many people as well as how music changes the body and mind during physical exercise. Specifically, music distracts people from pain and fatigue, elevates mood, increases endurance, reduces perceived effort and may even promote metabolic efficiency. Moreover, while listening to music, people run farther, bike longer and swim faster than usual – often without realising it.

There are a lot of debates about the advantages and disadvantages of listening to music during exercise. One of the main drawback is that rhythm of the music usually does not match the athlete’s pace. Also heart rate changes while listening to music, but whether the heart beats faster or slower depends on the tempo of the music. In the November 2009 issue of "Harvard Health Letter"[i] studies performed at Massachusetts General Hospital and in medical facilities in Hong Kong show that people who listened to music for 20 to 30 minutes each day had lower blood pressure and a slowed heart rate compared with those who did not listen to music. In addition to lowering blood pressure and heart rate - in some cases, when the music is slow - listening to music can reduce the perception of pain in some people.

Therefore, this led to the idea of current thesis. This paper describes a process of developing an application that will provide a user with the relevant song according to its cadence or heartbeat.

Project goals

The main purpose of this project is to develop an iOS application, which delivers a music that matches current user's heart rate or cadence in order to increase training performance. Specifically, it dynamically selects and play a song that suits user's tempo. But it also provides an ability to manually set up playlists in order to plan desired training pace.

Moreover, application will receive data from heartbeat sensors and accelerometers to detect cadence and give more accurate suggestions. It may conceivably improve training performance up to 10% according to many research papers described below.

General objectives

- Analyse local media-library and sort songs by bpm (beats per minute)

  Detect beats per minute of every song chosen by user. At every beat user will do a step or 2 steps if the song is slow. This can be called the rhythm of the song.

- Dynamically select a song according user's cadence or heartbeat.

  While running or training application should play music that suits user tempo. So at the moment of changing songs application should measure pulse or cadence and play song with corresponding rhythm

- Manually compose playlist according desired tempo

  Some users may want to plan workout and pre-define music to follow the desired tempo. So application should provide a mechanism for composing playlists from songs according its rhythm.

Contribution

The first part of the thesis describes system development process with applied methodologies and literature review of important background knowledge for developing the application. Market research is presented in Chapter 3. The system specifications and project requirements are presented in Chapter 4. Code structure and used tools are discussed in Chapter 5. It also contains programming logic, architecture and testing process. Chapter 6 contains thesis result, interface overview and conclusion.

Chapter 2. Development Process

System development process

The application development process has 3 stages: design process, system specification, iterative system implementation and documentation.

Design process

The five-stage model proposed by the Hasso-Plattner Institute of Design at Stanford[ii] was used as an approach to the design process. Specifically it is a Design Thinking Process(Fig 1.1).

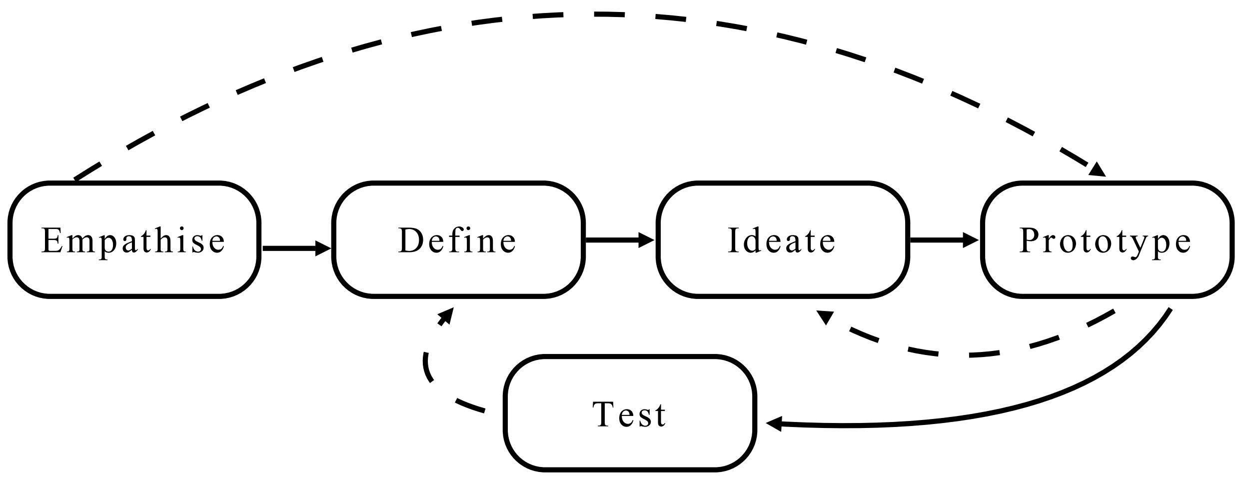


Figure 1.1 Five stages of Design Thinking Process

1. Empathising: Understanding the human needs involved.

   Many online music repositories provide special playlists for runners and athletes made by people. But they can’t cover all musical tastes and people can get bored with listening to the same songs every time. It also takes time to find suitable playlist.

Another type of services provides bpm detection. User upload or find music tracks and then compose playlists. It may take big amount of time.

Why do people listen to music?

Paula Radcliffe, the world record–holding marathoner, has said, “I put together a playlist and listen to it during the run-in. It helps psych me up and reminds me of times in the build-up when I’ve worked really hard, or felt good. With the right music, I do a much harder workout.”

According to a study published in 2015 by Jenny Groarke and Michael Hogan from the National University of Ireland[iii], people under 30 years old establish reasons for music listening like stress relief, bonding, and personal meaning. In older subjects, reasons included meditative effects, reducing loneliness, and novelty.

2. Defining: Re-framing and defining the problem in human-centric ways.

   People need to listen to the music they love.

   People need this music be suitable for running and training.

   People doesn’t want to spent much time on it.

   People need to listen to different songs.

3. Ideating: Creating many ideas in ideation sessions.

   The idea of creating an application that will make playlists according to the tempo. The application will analyse user's songs and detect cadence. It will match songs to cadence.

4. Prototyping: Adopting a hands-on approach in prototyping.

Before implementing a prototype, the mock-up was made (Fig. 1.2).

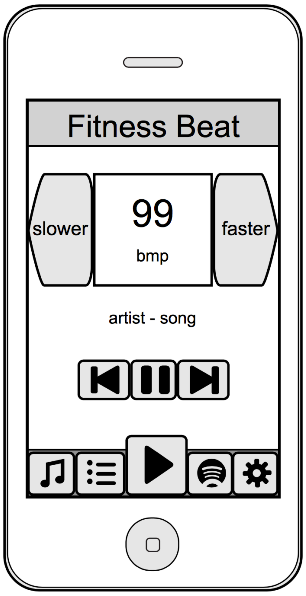


Figure 1.2 Early time mock-up

The first prototype(Fig. 1.3) has basic few features : detect bpm of chosen song, manually create a playlist, play and control music.

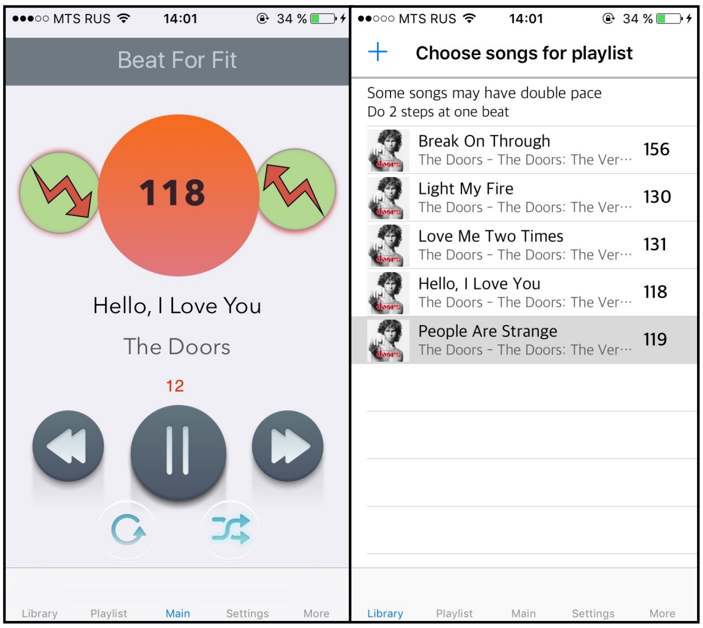


Figure 1.3 First prototype

5. Testing: Developing a prototype/solution to the problem.

    Empathic-design technique[iv] used to gain first-hand insights and learn things that are "difficult to explain" from a user prospective. Designer take a role of user. So the need of user interface improvements was revealed during testing prototype using this technique. And as a result of second iteration another prototype with improved user interface was made (Appendix 1). Also the name was changed due to copyright.

Background theory

The impact of music on training has been studied a lot by biomechanics, neurology, physiology and sport psychology disciplines. When people listen to the music they automatically tried to do all their movements according to the music tempo. Even heart rate adapts to the music rhythm. Different studies found that listening to music during doing sport or walking increase motivation, concentration and heart rate. But it works only for exercises in normal pace, peak activity is unaffected by music tempo[v].

Publications:

1. The University of Wisconsin – La Crosse

Year: 2003

Results: 15% increased productivity for volunteers who decided to listen to fast music by focusing on it. Also more power, higher heart rate and cycling faster.

Methods: 20 people listened 13 mixed songs. Volunteers cycling for 60 minutes at a pace and gear they prefer[vi].

Conclusion: Heart-rates rise from 133 to 146 BPM and power output increased correspondingly, when listening to the low tempo sound of ocean waves against music with a medium and fast tempo[vii].

2. The Research Institute for Sport and Exercise Sciences at Liverpool John Moores University

Year: 2009

Results: Heart rate and distance decreased with slow tempo. Greater distance and increased heart rate with fast tempo, in addition they liked the music more.

Methods: 12 people cycle at a pace that they could keep for half an hour and listening to their chosen songs. Next test, they rode the bicycles again, but the music tempo increased or decreased in different ways by 10% without notifying.

Conclusion: Though the subjects thought their cycling was harder at the faster tempo, the scientist found that when the music with fast pace was heard while exercising "the participants chose to accept, and even prefer, a greater degree of effort"[viii].

3. A group of researchers from Australia, Israel and the United States

Year: 2004

Methods: People run at a pace where they were at 90% of their peak oxygen uptake[4].

Conclusion: the music had no impact on their running pace or heart rate, regardless of the tempo of the music[ix].

4. A Questionnaire Study among Athletes.

Year: 2011

Results: Athletes listen to music to concentrate on trainings and performance often during pre-event, pre-training sessions, and warm-ups.

Methods: Questions relating the empirical motives for listening to music

Conclusion: Athletes feels that listening to music increase activation, positive affect, motivation, performance levels, and flow[x]. Moreover, there are some types of workout music that use brainwave entrainment which pretend to boost performance.

5. New York Times and Max Planck Institute

Year: 2013

Methods: Group of people tested twice: first they working out in normal setting, and second they trained with machines that integrate beats and rhythms into each repetition. The kits were installed into stair-stepper and two weight machines with bars which could be pushed or pulled down to activate different muscles. During every workout, the researchers measured the force participant generated while using the machines, and the weightlifters' movements tended to stutter or flow. They also control oxygen the participants consumed, a valid measure of physical effort. Then, the researchers asked the participants to put a mark from 1 to 20 of the tolerability or unpleasantness of the workout.

Conclusion: "Participants could express themselves on the machines by, for instance, modulating rhythms and creating melodies." said Thomas Hans Fritz[xi].

Chapter 3. Market research

Target market

According to the statistics provided on statdata.ru[xii] and appleinsider.ru[xiii], it was implemented rough target market segmentation(Fig 1.4). Based on it the presumable number of potential customers was calculated. Constantly narrowing down towards initial target customer gave result of about 10 million people who represent BeatForFit target market.

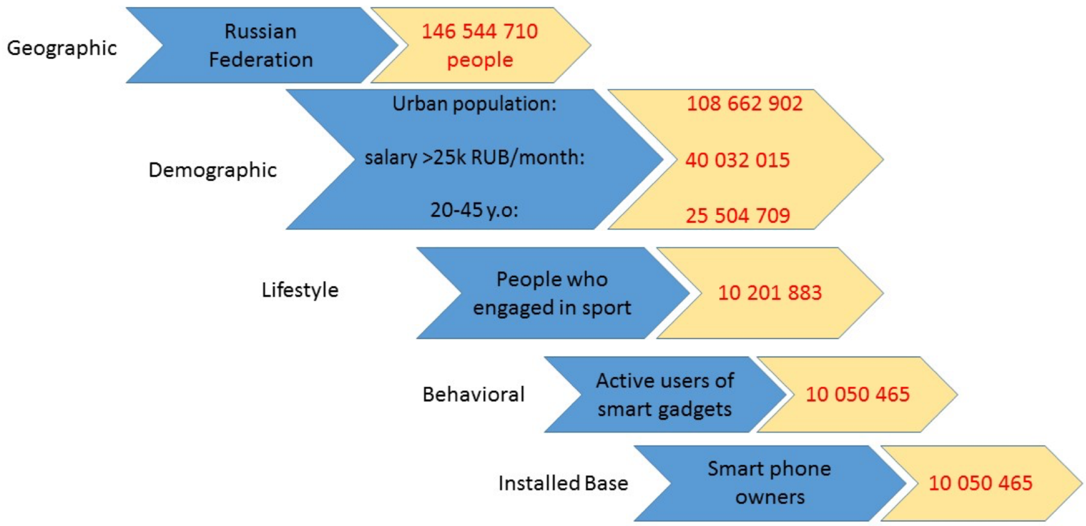


Figure 1.4 Target market segmentation flow chart.

SWOT Analysis

The SWOT Analysis of the business idea was undertaken as one of the market research tool. Particularly, it defined the objective of the business and identified the internal and external factors, which are essential for achieving the business objectives. Table 1.1 demonstrates all aspects that help to realise the strengths, weaknesses, opportunities and threats involved in the business.

Table 1.1 Business aspects

Strengths	Opportunities
• Innovative product in Russia • Generates playlist according user's tempo • Listen to local songs • Improves training performance up to 10% • Functionalities are available offline	• After strong establishing on the Russian market open to foreign branches • Distribute the product worldwide
Weaknesses	**Threats**
• Available exclusively on iOS	• Probability of competitors' occurrence • Existence of substitutes.

As it can be observed weaknesses and strengths of the product are internal factors to the business, whereas opportunities and threats are external ones.

Industry Sector

Technavio analysts forecast the global fitness app market to grow at a CAGR of 31.35% during the period 2016-2020[xiv]. Revenue in the global market fitness applications will be about 670m $. And it will rise to 2.62 billion by 2020(Fig. 1.5).

Russia has 2-3% of world revenue and it means that by 2020 market will be about 52m $. It is not much, but it should be considered that an average application costs 5$. People spent more on devices and services.

According to experts, the key incentives for the active growth of the market are:

- The high penetration of smartphones and the Internet
- The availability of cost-effective application
- The growing demand for fitness apps among women

It seems to be nothing special, but the amount of these incentives lead to a powerful development of the market.

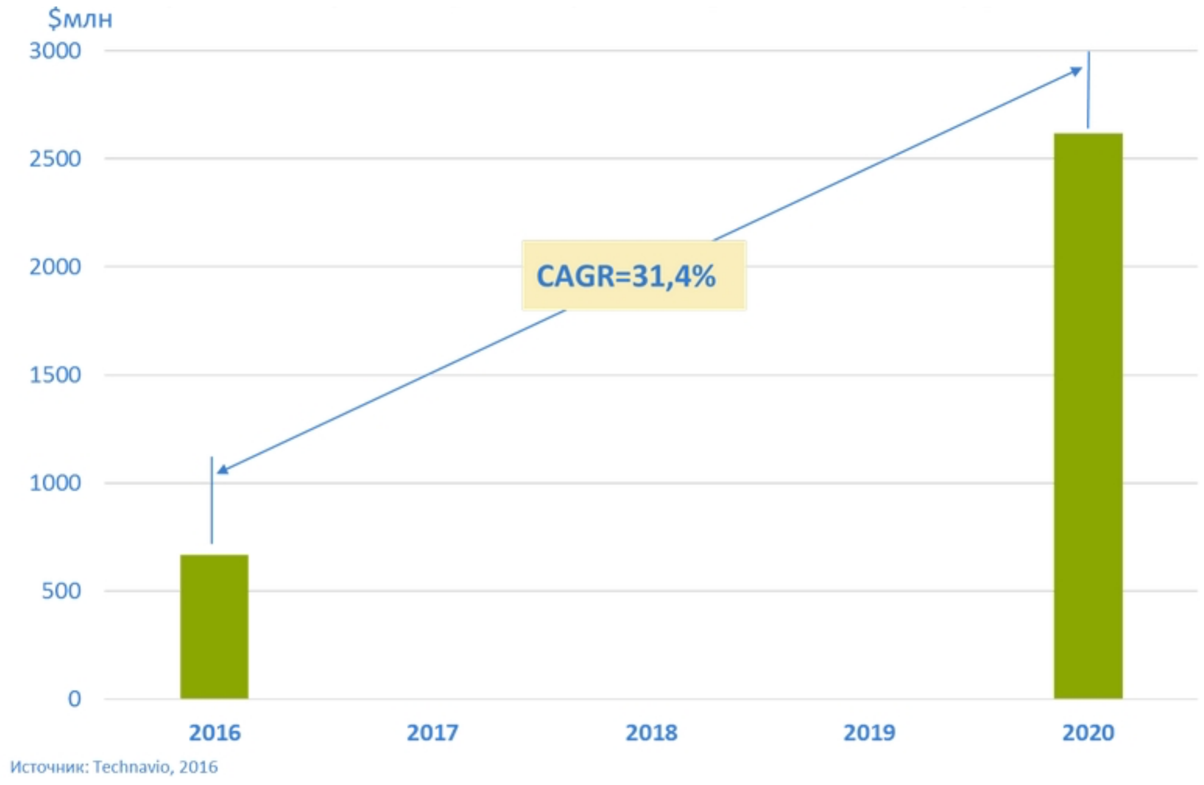


Figure 1.5 The forecast of the world market of fitness applications

Another interesting thing is that mobile fitness applications is a part of mHealth market and it is the biggest part - about 36%(Fig. 1.6).

In 2013, the use of applications in the category of health and fitness has grown by only 49%, with an average growth in the industry at 115%. But 2014 has brought a dramatic change - application for fitness are growing twice as fast as the market average, increased their attendance by 62% in six months, with an average growth of 33%(Fig. 1.7).

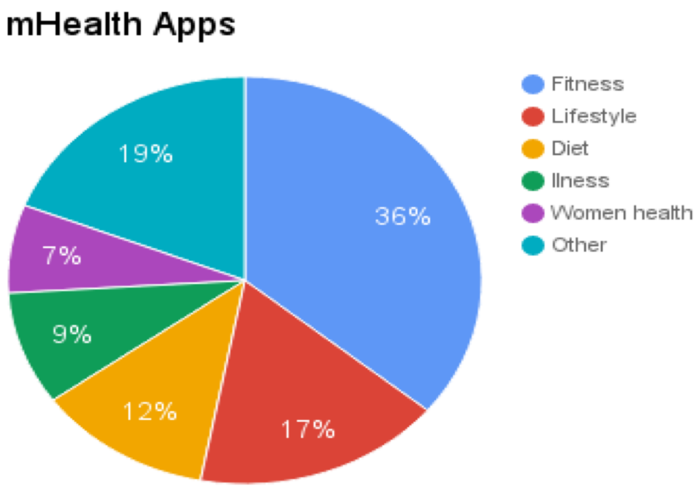


Figure 1.6 Types of mobile medicine(mHealth) applications.

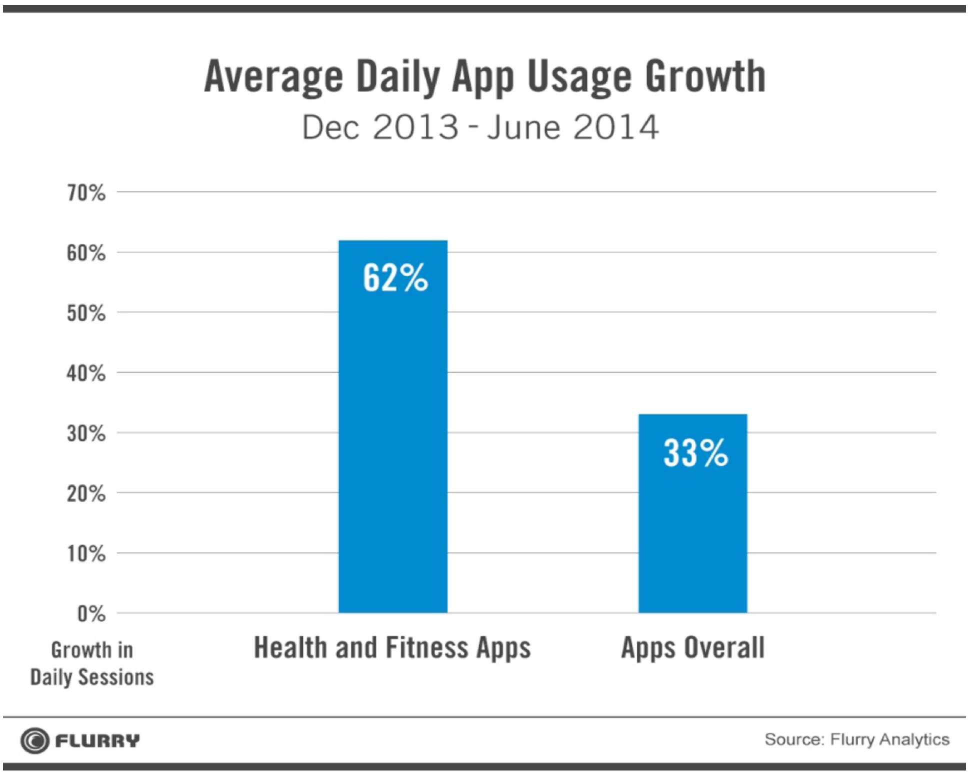


Figure 1.7 Average daily app usage growth

Competitive environment

According to Coupland-Smith[xv], a competitive environment is the dynamic external system in which a business competes and functions. The main and direct competitor of BeatForFit application is Spotify Running. To compare these two software applications, Spotify Running has range of disadvantages such as:

- Can be used only through the Internet
- Only Spotify's song collection is available
- Lack of opportunity for users to create their own playlist
- Expensive cost
- Can't set the aim tempo
- Only for runners
- Does not support fitness bands

As for indirect competitors of Fitness Beat, they are RecordBeater, Second Wind, RockMyRun, Jog.fm. Most of them have the same drawbacks as Spotify Running.

In fact, availability of such substitutes ensures strong competitive environment in the industry. However, competitive advantages of BeatForFit which were described in the SWOT analysis make it unique and more useful than other applications.

Key vendors

- Azumio
- Fitbit
- Jawbone
- Runkeeper
- Under Armour

Other Prominent Vendors

- Adidas
- Daily Workouts
- Google Fit
- Nike
- Noom
- Polar Electro
- Runtastic
- Samsung Electronics
- Sports Tracking Technologies
- Wahoo Fitness

Market driver

- High penetration of smartphones and Internet

Market challenge

- Existence of low-quality apps

Market trend

- Increase in number of mergers and acquisitions

PESTLE analysis

PESTLE is a macro environmental framework used to understand the impact of the external factors on the product and is used as strategic analytical technique.

P - political forces

- Politics of healthy way of life in all developed countries
- Government build special paths for running and cycling, creates parks and make sport events (marathons, games and competitions). This will help to promote application.

E - economic factors

- People spend more and more money on health and gadgets
- Growth of mHealth market despite the economic crisis
- Decrease of quality of life in some countries causes people to spend less on non-critical needs like sport and music equipment (including mobile applications).

Since BeatForFit is a free sport application people can use it without spending money that meets their needs.

S - socio-cultural factors

- Trend of healthy lifestyle
- Gadgets become more popular
- People of all ages try to go jogging
- Social ads that encourage people go for sport

The application is using fitness bands and made primary for runners, so it is fully in trend.

T - technological factors

- Technology has a short lifetime
- Fitness bands upgrades fast
- iOS gains new features and SDK changes with versions

BeatForFit needs to upgrade often, adapt new bands and continuous development process. That means permanent developer team and distribution through the App Store.

L - legal factors

- The application uses music from local library so it not under the regulation of Copyright Act - it only plays music not download or store
- It uses free-to-use (open source) SDKs and API
- In some countries it is prohibited to cycle in headphones

E - environmental impacts

BeatForFit is a mobile application so it has not impact on environment.

Chapter 4. System specifications

Specific objectives

- Detect user cadence or heartbeat.
- Use user local media as a source.
- Calculate bpm of chosen by user songs.
- Match songs with athlete cadence.
- Play music that suits user's heart rate
- Manually created playlists
- Remote music control.
- Support fitness wristbands and Apple Watch
- Detect cadence using inbuilt accelerometer

Assumption, constraints and risks

Effective planning is very important for project. To reach the project goals the good way is to define assumptions, constraints and risks. (Table 2.1) It provide a metric to control changes.

Table 2.1 Assumption, constraints and risks

Assumptions	Constraints	Risks
1. People will use the application while training 2. Application will help athletes to improve their training performance 3. The application will generate playlist that suits user's tempo	1. Only iOS user will be able to use the application 2. Some models of smart watches wristbands will not support the application 3. There will be not enough songs in user's local music library to satisfy desired tempo	1. There might be a huge amount of not iOS users desiring to use the app

Product requirements

User stories was used as requirements gathering technique (Table 2.2). Since this technique provide flexibility and the description of features in natural language. For user story creation a "Three Cs" formula was applied. The formula is the "Card, Conversation, Confirmation" model that was proposed by Ron Jeffries[xvi] to distinguish informal user stories from formal requirements practices such as use cases.

Roles:

USER - a person, who use the application on iOS.

Table 2.2 User stories

<table>
<tr><td colspan="3">High priority</td></tr>
<tr><td rowspan="3">1</td><td colspan="2">Count and store bpm of tracks</td></tr>
<tr><td>As a USER
I want to know the bpm of my songs in the music library
in order to create a playlist of my planning workout</td><td rowspan="2">Success
a) The algorithm counted the bpm of the music tracks and saved the result.

Failure - log the message:
a) “Bpm cannot be counted, the track will be not included into the library”</td></tr>
<tr><td>Bpm is being counted only once, when is being added into the library, and then stored inside the application.</td></tr>
</table>

<table>
<tr><td rowspan="3">2</td><td colspan="2">Manage playlists</td></tr>
<tr><td>As a USER
I want to be able to manage my playlist
in order to plan my workout</td><td rowspan="2">Success - playlist successfully created or deleted.
Failure on creation - log the message:
a) “The playlist can't be created”</td></tr>
<tr><td>“Music library” is treated like a usual playlist and is the default one.</td></tr>
<tr><td rowspan="3">3</td><td colspan="2">Play music with bpm corresponding to heart rate</td></tr>
<tr><td>As a USER
I want to listen to the music that suits my pace
in order to run better</td><td rowspan="2">Success - the heart rate is measured and the playlist adjusted to it.
Failure - display message.
a) “Cannot find the fitness band. Please, specify the bpm manually”</td></tr>
<tr><td>Heart rate is being measured by the fitness band.</td></tr>
<tr><td rowspan="3">4</td><td colspan="2">Play music with the specified bpm</td></tr>
<tr><td>As a USER
I want to choose the music tempo
in order to run with defined bpm</td><td rowspan="2">Success - the playlist filtered with the specified bpm.
Failure - if there are no such tracks, the application plays the closest ones and shows the message, if there are no close enough tracks.
a) "Add more songs"</td></tr>
<tr><td>Filter the playlist using the bpm, specified by the user.</td></tr>
</table>

5	**Player controls**	
	As a USER I want to have usual functions of managing music playing process in order to have the ability to control the playback as I used to in other music players	Success - functionality works correctly: “play” starts playing, “stop” - stops, “next track” and “previous track” - switches playback to the next and previous one correspondingly. Failure: a) Local file is not available, switch to the next one in the playlist.
	Functions are: play, stop, next track, previous track	
6	**Fitness band compatibility**	
	As a USER I want my fitness band work with the application in order not to buy special devices	Success - user’s fitness band is compatible with the application. Failure: a) Show notification “Your fitness band is not supported by Apple Health app” and switch the user to cadence mode.
	Famous fitness bands should be compatible with an application.	

Low priority		
	Statistics	
7	As a USER I want to have the stored statistics (date of training, amount of steps, duration, average bpm) in order to see my progress	Success - the statistics is stored and shown correctly. Failure: a) If fitness band is not available, save null amount of steps and show it as empty value, when displaying statistics.
	Amount of steps is being calculated by the fitness band.	
	Map of the workouts	
8	As a USER I want to see the map of my workout in order to analyze my routes of previous trainings	Success - map is shown to the user. Failure - display the message: a) “The map is unavailable since there is no internet connection”
	Store coordinates of the route locally, show the map, using a web service.	

Chapter 5. System implementation

Tools

There are two dominant operating systems for smartphones: Android by Google and iOS by Apple. According to statistic made by Gartner, Inc. both this two OS has 99.6% of the worldwide smartphone sales to end users in the fourth quarter of 2016[xvii]. The final outcome was developing for iOS mobile operating system due to available resources and author's preferences.

The system development process consists of:

1. Making a mock up at moqups.com

This is a free online tool that provide quite a lot instruments but in the same time it is easy-to-use for the author's purpose.

User interface design using Sketch 3.

Sketch is a vector design tool primary for making user interface design. It is required special knowledge to work with the program, but provide high performance and powerful set of instruments to work with graphic.

Coding in xCode integrated development environment.

xCode is native tool made by Apple Inc. for developing programs for iOS and MacOS. It provides interface builder, debugger and testing tools. At the final stage version 8.3.1 were used that support the latest iOS and Swift 3 programming language updates.

Swift 3 was chosen as a programming language since it is modern and intended to be more resilient to erroneous code than Objective-C (another programming language for Apple products), and more concise.

MacBook Pro with MacOS Sierra and iPhone SE with iOS 10 were used during the whole system development process.

Logic

The system is consist of 3 core logic modules: Song analyser, music player and motion manager.

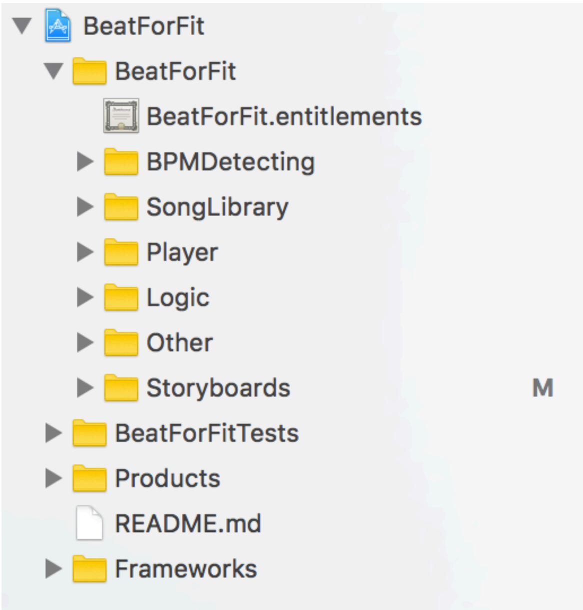


Figure 3.1 Program code organisation

Song analyser logic module is responsible for detecting bpm (beats per minute) of every user chosen song.

Music player logic module is responsible for playing music and provide user a way to control songs remotely via control buttons on headphones, speakers or from lock screen without a need of unlocking device. User can play next/previous song, play/pause music and play a song with faster/slower tempo.

Motion manger logic module is responsible for retrieving and handling health data - pulse and cadence. It uses API provided by Apple and consists of two modules: pedometer and heart rate manager.

Architecture

The iOS mobile application follows Model View Controller meta pattern recommended by Apple (see fig. 3.2).

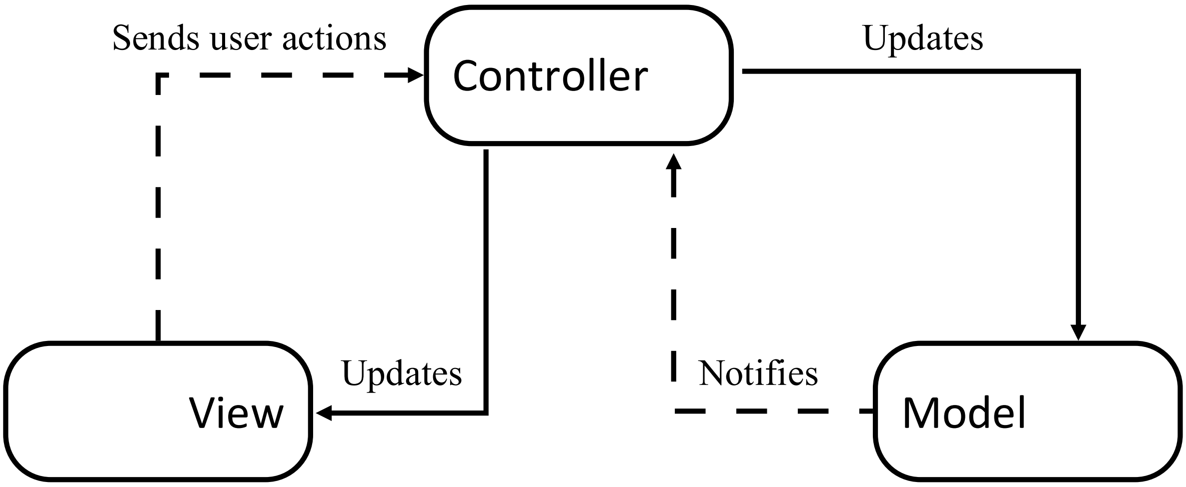


Figure 3.2 Model View Controller architectural pattern.

The code consists of 21 classes with 7 extensions, 6 protocols and 3 structure (Appendix 2). In total it is more then 2000 lines of code (Table 3.1).

Table 3.1 Total lines of code

Language	Files	Blank	Comment	Code
Swift	26	642	790	2059
JSON	22	149	0	12119
Markdown	1	2	0	6
SUM:	49	793	790	14184

Classes Storage, Pedometer, HeartRateManager, Player and SmartSongSelector are following singleton[xviii] architectural pattern in order to keep only one single instance during runtime.

Classes Pedometer and HeartRateManager have delegates. The delegation pattern[xix] is a design pattern that provide a way to reuse a code without inheritance. Specifically, in the system it delegates the function of updating pulse and cadence values.

Classes description

For persistently storing values and objects a singleton class Storage uses NSUserDefaults object. The NSUserDefaults class is a part of Apple's Foundation framework that provides a API for interacting with the defaults database system. It stores an array of Song type instances and some dictionaries and arrays for fast access (e.g. ordered array of BPM values).

Type Song is internal type for storing songs in runtime. It has such properties as name, artist, album, bpm, URL, image, id and can be easily modified in future. All variables are optional, so it could have a value or not. Swift programming language provide powerful mechanisms for working with optional variables.

Class SongPicker is conforming to MPMediaPickerControllerDelegate protocol. It is responsible for providing user an interface to choose songs from local library and initiate a process of detecting BPM.

Classes TempiBeatDetector, TempiFFT, TempiPeakDetector, TempiDSP and TempiUtilities are parts of open-source system. The main purpose of this system is to detect BPM of audio files. It is based off of the autocorrelation algorithm[xx]. The essence of autocorrelation is to detect a degree of identity between a signal and itself at a certain lag. For the the BPM detecting audio file is divided to many samples. Then look through the signal by taking every sample and execute a correlation between reference window and the lagged window. The correlation zero lag is the global maximum due to collating the reference to a direct copy of itself. As it performs further, the correlation is certainly decrease, but in the case of a periodic signal, at some point it starts to increase again and then achieve a local maximum. The distance between zero lag and that first peak is an estimate of tempo. The system is no longer available online but it was under MIT License[xxi].

CellForLibrary class and .xib file is responsible for representation of Song instance in Library and Playlist tables. It follows the iOS Human Interface Guidelines[xxii] and looks similar to native music player.

ProgressBar class is responsible for instantiating UIProgressView from native UIKit framework. It represents the current progress of analysing songs on a scale of 0-100.

Class Player conforms to AVAudioPlayerDelegate from AVFoundation native framework and responsible for playing songs. The instance of type Song is passing through the argument by SmartSongSelector. For playing playlist it receives an array of internal URLs.

Class SmartSongSelector is responsible for choosing a song to play. Depending on the current mode(manual/cadence/pulse) it selects a song from global storage. If "cadence" or "pulse" mode selected it is guaranteed that it would be no repeated songs. There is a copy of global song indexes and after playing a song its index removed. This copy is reset every 30 songs or 3 days. Depending on mode it takes a current value of desired bpm, cadence or pulse and selects a song with corresponding tempo. If there no exact tempo matching, then performs a findClosestBpm function. The idea is to provide a user with continual stream of music.

Pedometer class is responsible for retrieving data about user current cadence. It has a CMPedometer instance that is part of native CoreMotion[xxiii] framework by Apple. The Core Motion framework provide an API for receiving motion data from device hardware and process that data. Pedometer class use delegation for updating "cadenceLabel" for displaying current cadence to user and set global variable "cadence".

Heart rate manager is responsible for getting user pulse and it works with native HealthKit[xxiv] framework by Apple Inc. HealthKit enables iOS developers to integrate health and fitness devices with their app and integrate the data with

Apple's easy-to-read dashboard. HealthKit let health and fitness applications on an iOS device to read shared health and activity data from native Health application. Health is an inbuilt application made by Apple that contains all user heath and activity data or statistic. A number of companies support HealthKit, including Polar, EPIC, Mayo Clinic, and RunKeeper.

Semaphore class is used for controlling executing bpm detection function in multithreads. So it is guaranteed that BPMs would not be mixed.

Class BpmDetector is responsible of interacting with TempiBeatDetector interface. It received array of chosen songs and proceed it song-by-song for bpm detection. The outcome is the same array but every song has calculated bpm.

User interface implementation

User interface design in Sketch include icons drawing, colour scheme and layout. During the design phase it was stated that application should provide a friendly interface.

People will use application during doing sport, so all the buttons should be big enough. Function buttons "speed up" and "slow down" have a text caption and corresponding icons to clearly represent the meaning (Appendix 1). Labels "current bpm" and "current cadence/pulse" are also have informative functions. They give a tips to user like "start running" or "select cadence to start", that way user will learn how to use application. But the best way to show user how to interact with application and explain how it work it is mobile application onboarding[xxv] through interactive instructions. it will be implemented in future release.

Icons of tabs on TabBar have 2 states: active or not. According to Apple Human Interface Guideline[21] icon should be the same but highlighted or filled when active. Some of icons were downloaded from free legal repositories and modified.

Application has a style that every user interface entity should follow. To achieve it Google Material Design[xxvi] concept was used. The colour palette has 4 colours and every element is one of this colour (Fig. 3.3).

Figure 3.3 Colour palette

All sights and text in San Francisco font. It is a new system font for Apple devices. The only difference is in spacing.

The application icon symbolised a phrase "listen to your heart" and represent one of the main feature - match songs to pulse. Apple Store requires icon 14 different sizes, 2 of it demonstrates below(Fig. 3.4).

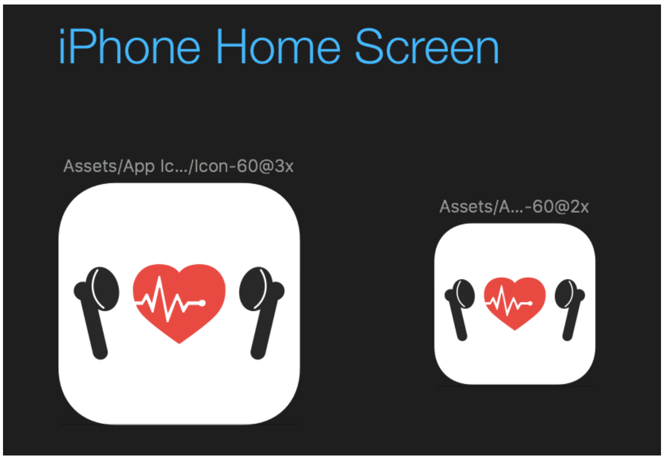


Figure 3.4 Application icon.

Since application design was built according guidelines and use system fonts it harmoniously fits into the iOS environment.

Testing

The main purpose of testing is to verify that the code meets its specifications and to improve software quality. For the system testing 3 types of testing were performed: functional testing, non-functional and usability testing.

Functional testing refers to activities that verify a specific action or function of the code. Functional tests tend to answer the question of "can the user do this" or "does this particular feature work." To verify that particular feature work user stories (Table 2.2) were used as a functional point. Low priorities user stories are not implemented yet, so no test needed.

Non-functional testing refers to aspects of the software that may not be related to a specific function or user action, such as scalability or other performance, behaviour under certain constraints, or security. Testing will determine the breaking point, the point at which extremes of scalability or performance leads to unstable execution. Non-functional requirements tend to be those that reflect the quality of the product, particularly in the context of the suitability perspective of its users. During this kind of testing the most performance demanding process was discovered. It is BPM detection.

There are only 3 SDKs for iOS that compute bpm of music and very few discussions into the web. Also there is no performance comparing articles available online. So the decision of making performance and accuracy testing were made.

The fist SDK was the most famous one - The Amazing Audio Engine[xxvii]. The tests showed that bpm detecting is incorrect for some songs. Since BPM detecting is the core feature of the system it should be perfect. The Amazing Audio Engine has been retired for the moment of writing paper.

The next one is Superpowered Audio SDK[xxviii]. It is a software development kit based on Superpowered Inc's digital signal processing (DSP) technology. This SDK showed great performance but accuracy was also bad.

And the final is TempiBeatDetection[xxix]. It is a library that performs real-time or static beat detection on audio and written in Swift. The performance tests showed normal results but accuracy is great. So TempiBeatDetection was chosen to use.

Usability testing is to check if the user interface is easy to use and understand. It is concerned mainly with the use of the application. For performing usability testing Hallway testing method was used. Hallway testing is a fast and easy method of usability testing in which randomly-selected person is asked to use the system. So 5 students were asked in different situation. Some of students reports that interface is unclear. Based on it some improvements were made: buttons with the description of functionality and more direct instruction.

Chapter 6. Results

Thesis outcome

Thesis outcome is training performance aware music recommendation system. The system is a mobile application for devices running iOS operation system. The application has following features:

1. Detect music tempo
2. Play music from local library
3. Create playlist
4. Play music that match cadence
5. Play music that match pulse
6. Remote music control
7. Songs does not repeat during training
8. Manually set the desired music tempo
9. Support fitness bands

Interface overview

On the first launch of BeatForFit the application music library is empty. User should go to the "Library" tab and choose songs of analysing (Fig. 4.2). If user push any music control(play, next, speed up etc.) button then the pop-up window will appear asking user to add songs(Fig. 4.1)

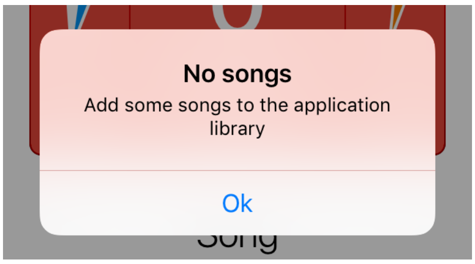


Figure 4.1 Pop-up window

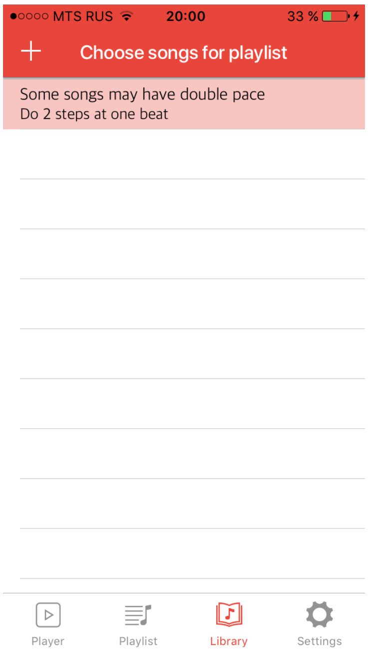


Figure 4.2 Library tab

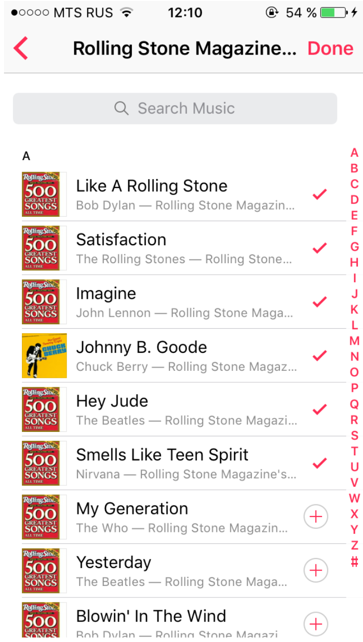


Figure 4.3 Media picker

To add songs to the application library user should push "+" on the upper left corner. The music media picker appears with all songs stored on local device (Fig. 4.3). The process of BPM detecting initiates when the user tap "Done". Some songs will have double BPM value and there is an information on the top of Library tab that user should do two steps for one beat.

Informative window appears (Fig. 4.4) and the array of chosen songs start to analysed. To indicate a progress of this resource demanding process there is a progress bar above a table of songs (Fig 4.5). The bpm value appears as soon as song processed.

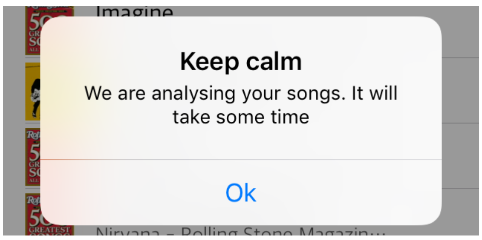


Figure 4.4 Informative window

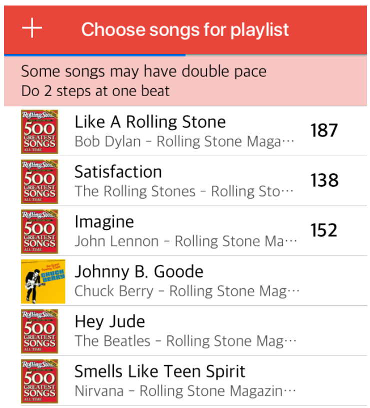


Figure 4.5 Playlist tab during the process of BPM detection

There are four modes of playing music:

1. Create playlist with desired bpm

2. Choose music tempo manually

3. Play songs that match cadence

4. Play songs that match pulse

User can add song to playlist on "Library" tab by tapping on it (Fig. 4.6).

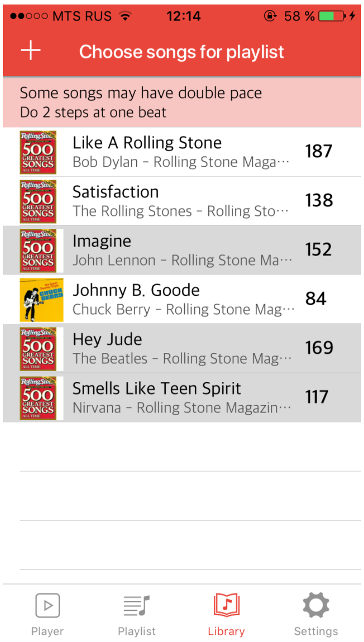


Figure 4.6 Song selection for playlist.

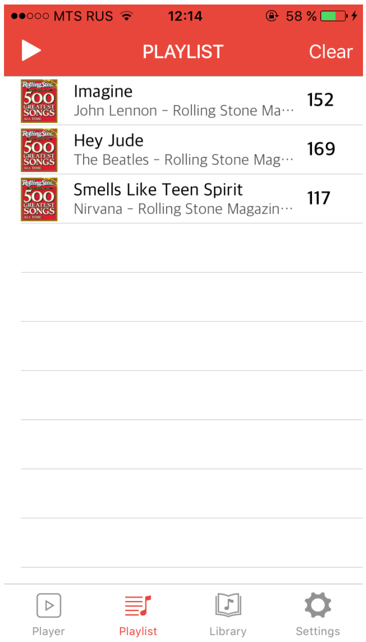


Figure 4.7 Playlist tab

On the playlist tab there are three functions:

- Clear playlist on the top right corner
- Play playlist on the top left corner
- Delete a song by swiping a cell to the left

By tapping a Play button music start playing in playlist order.

Player tab is the initial and main view. Here user can control music and set playing mode (Fig. 4.8). There are 5 music control buttons: play/pause, next song, previous song, speed up and slow down music tempo. The info section in the middle display the current BPM of the song and information depending on selected mode: if it is "Cadence" or "Pulse" mode then the current cadence or pulse displayed. If there is no cadence or pulse data available in HealthKit then instructions displayed (Fig. 4.9) when the data becomes available, it appears on the screen and song with corresponding bpm starts playing. If there are no songs with corresponding BPM, then the closest BPM value picked. In "Cadence" and "Pulse" modes smart song selection mechanism decided what snog should be played. It play songs in different order every time and guarantee the diversity by excluding recently played songs.

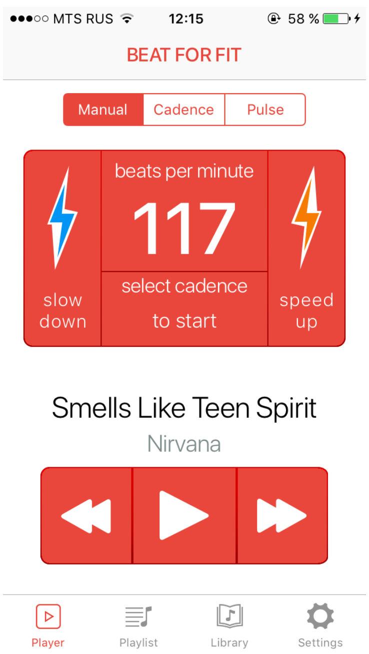


Figure 4.8 Player tab

By tapping on "play" button the song with corresponding to BPM label value starts playing. "Play" button becomes "Pause" and change picture. By touching on "Pause" button music paused.

"Next" button can perform depending on source of songs. If currently playing music is from playlist, then pressing "Next" button will play next song in playlist. If it is manual mode, then next song with the same BPM will play. If there is no such song, then the next BPM value (in increasing order) will set and play. If it is cadence or pulse mode, then smart song selector will choose a song according current cadence/pulse value.

"Previous" button plays previously played song regardless of the mode.

"Speed up" and "Slow down" buttons change the BPM value to the next/previous one and play corresponding song.

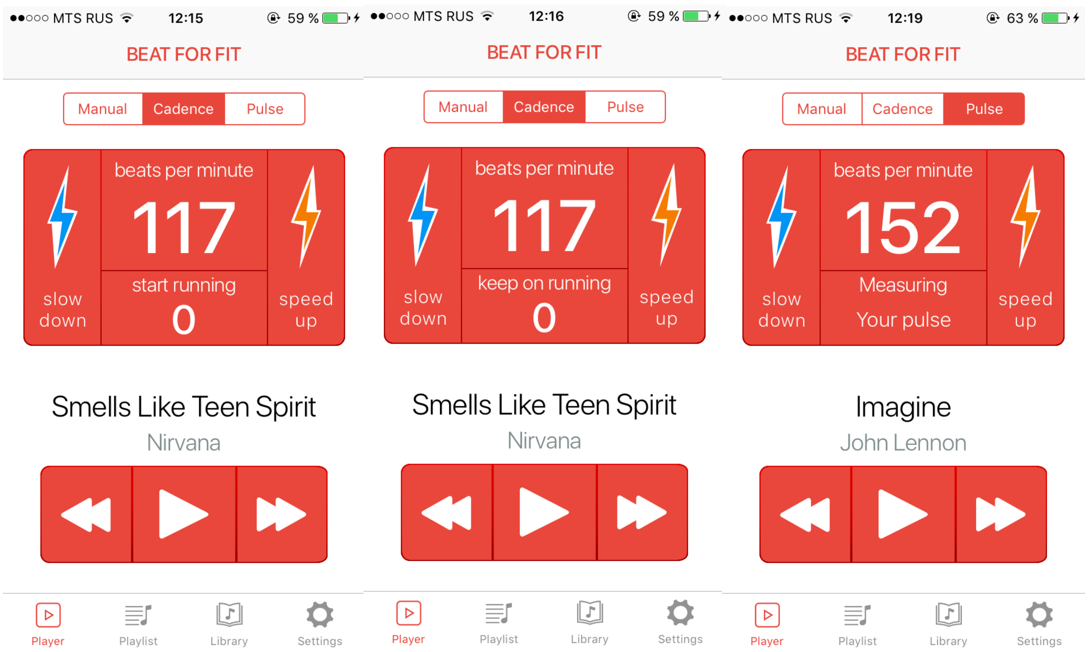


Figure 4.9 Player tab with various selected modes

The default mode is manual. On the top of the "Player" tap there is a selection bar with 3 modes.

In manual mode user set and change the music using control buttons described above.

In cadence mode application play songs according current cadence value. If the cadence value changed music will play till the end of song and then smart song selector will play a song according to the cadence value on a moment of changing.

The idea is that music should not stop. If there is no songs with corresponding BPM then the closest BPM value will set and play(Fig 4.10).

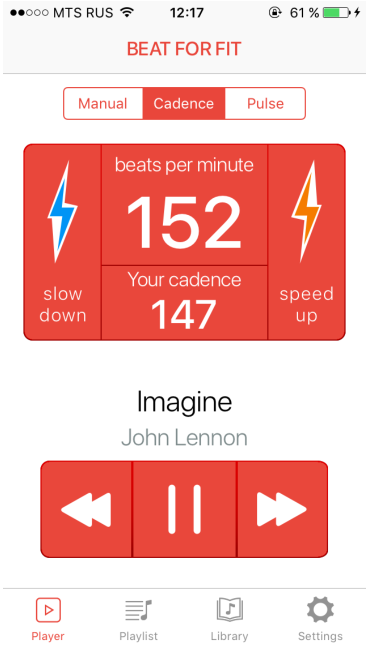


Figure 4.10 Player tab with selected cadence mode

When the first time application access fitness data the request dialog display asking user to provide a permission to read data(Fig. 4.11).

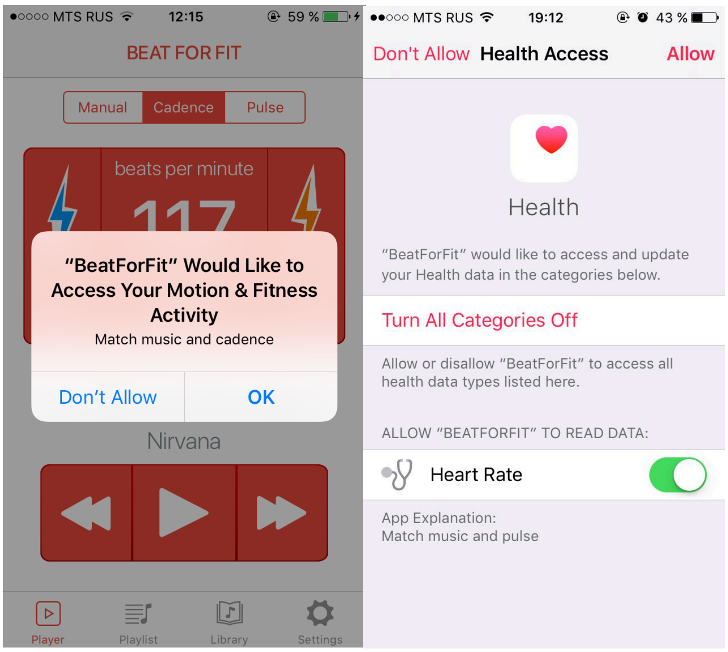


Figure 4.11 Permission request dialog.

The last tab is settings. For the current moment it has only one button - "Reset all alerts" which set the alert displayed state to the initial one (Fig. 4.12). But this view will be filled in future.

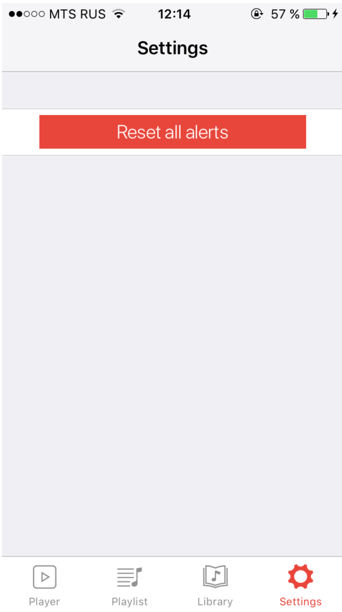


Figure 4.12 Settings tab.

Future work

At the current state application is tested and work properly. But for the future development there are a lot of features to add (Table 4.1). The system has a good option for enhancement since it combines pedometer, fitness tracker and music player.

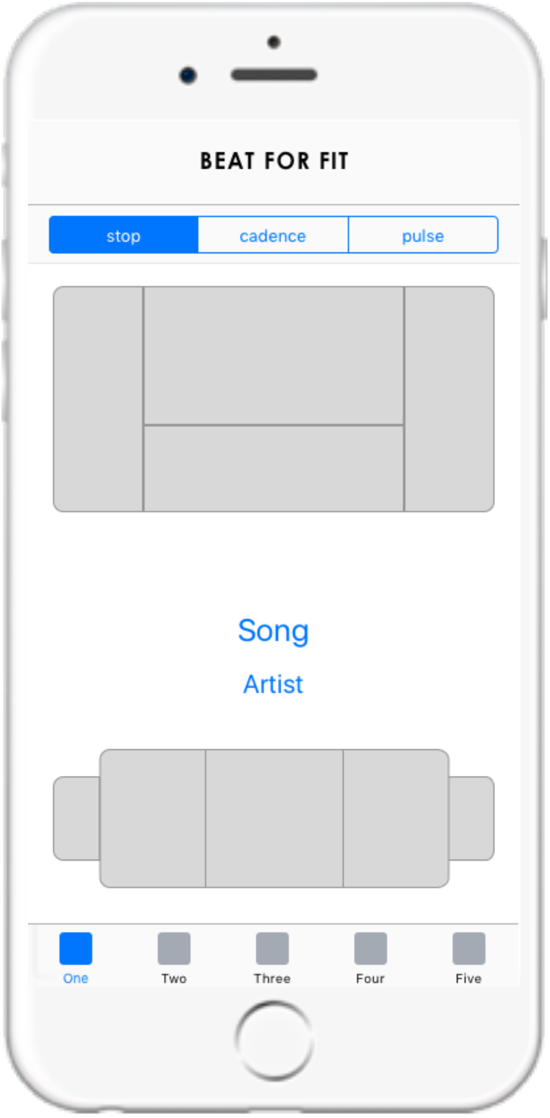
Table 4.1 Features to add

N	Feature description	Estimation time, hours
1	Possibility to have several stored playlists	30
2	Automatically compose playlists based on desired speed	50
3	Spotify integration	160
4	Workout map and fitness statistic	140
5	Pre-defined training plans	50
6	Warm-up and boost songs	20
7	Improve performance	40

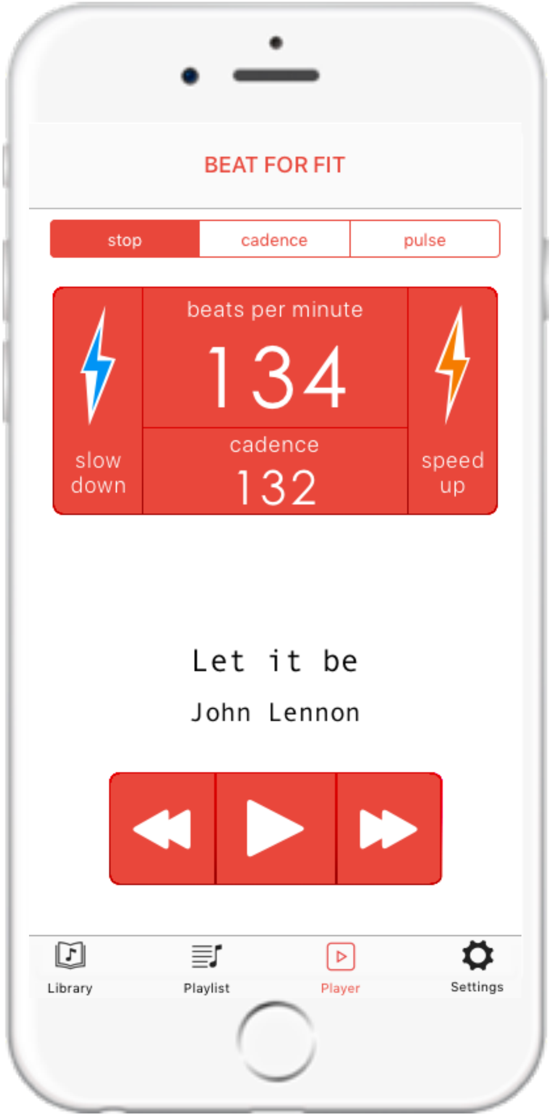
Conclusion

The idea of suitable music for running and training is quite popular and the developed application has big potential due to customer research. People love the interface and demonstration but never tried it in real life. BeatForFit will appear in App Store soon and collect real customer feedback. Based on this the application will be further developed and may be commercialised. Time will show how good is this business.

Appendix 1



Prototype

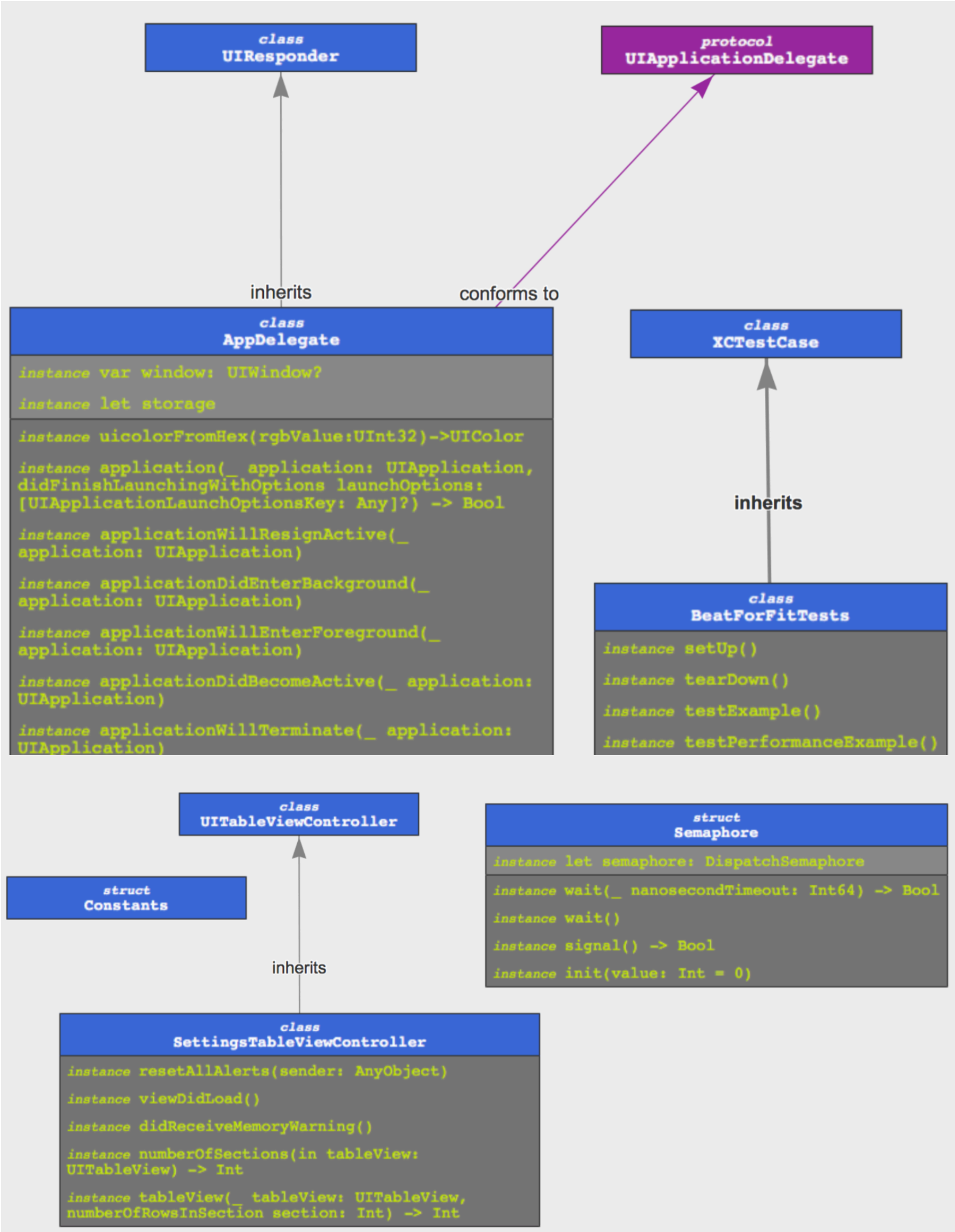


Design

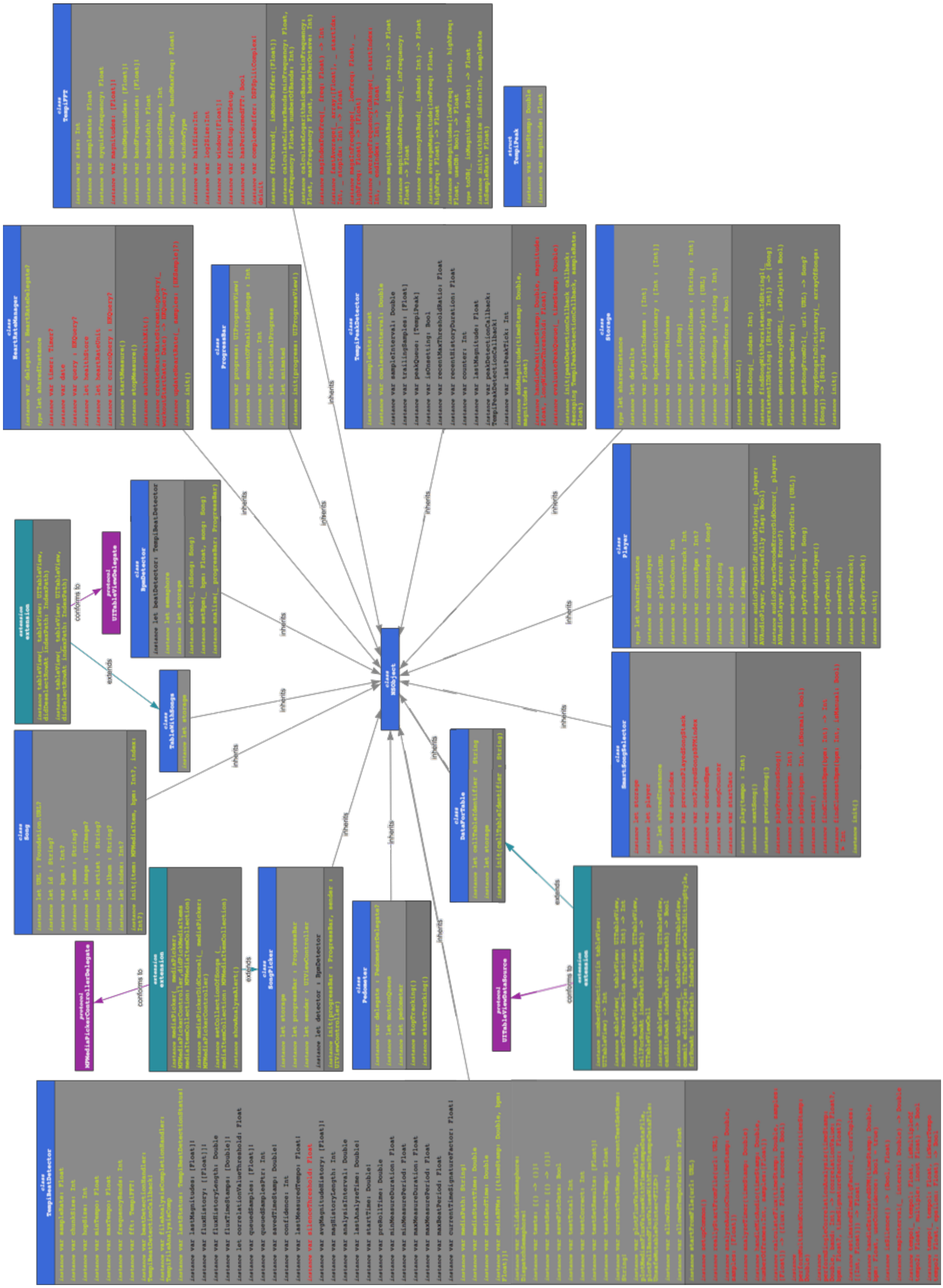
Palette

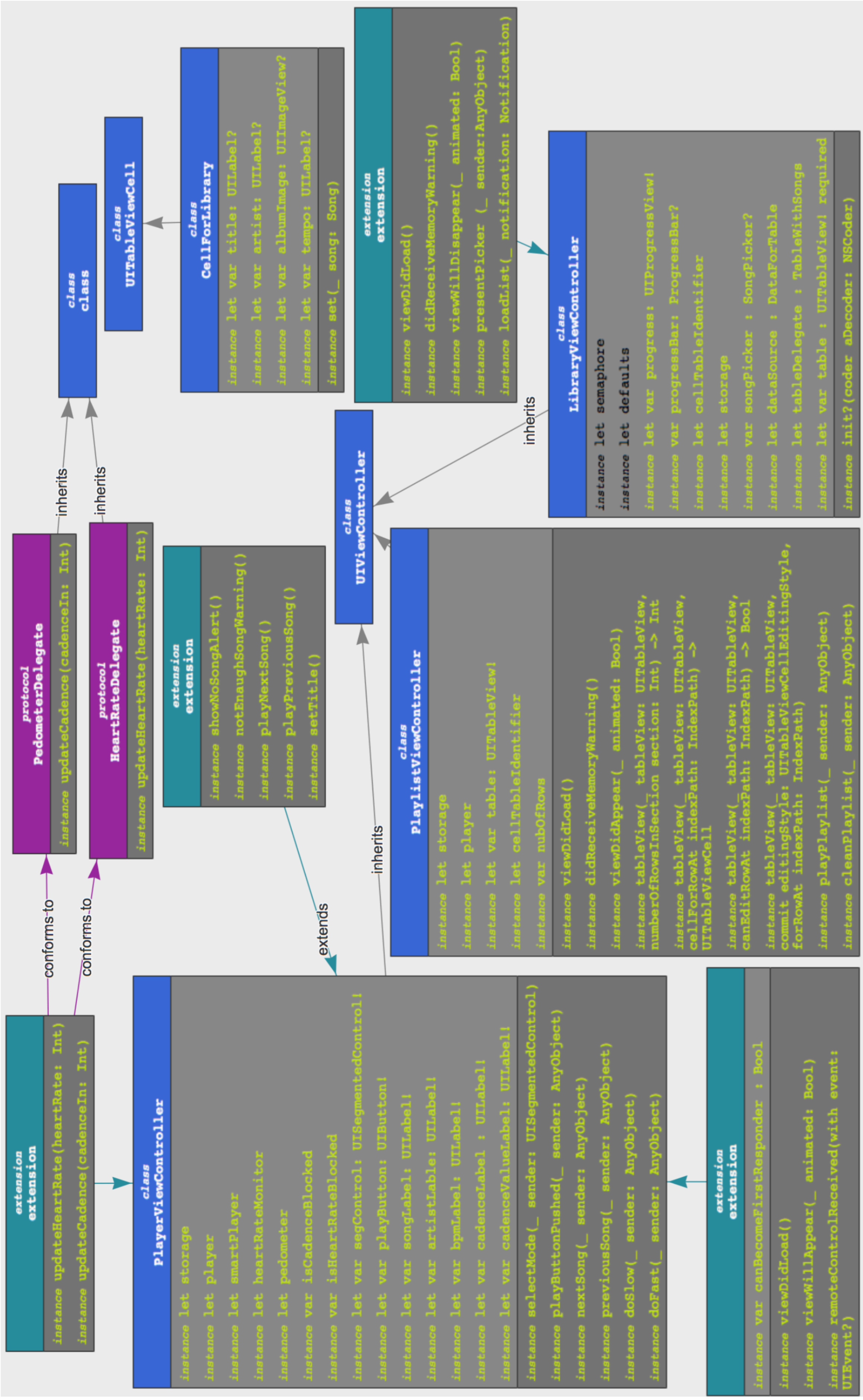
Icons

Appendix 2



class
UIResponder
protocol
UIApplicationDelegate
inherits
conforms to
class
AppDelegate
instance var window: UIWindow?
instance let storage
instance uicolorFromHex(rgbValue:UInt32)->UIColor
instance application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool
instance applicationWillResignActive(_ application: UIApplication)
instance applicationDidEnterBackground(_ application: UIApplication)
instance applicationWillEnterForeground(_ application: UIApplication)
instance applicationDidBecomeActive(_ application: UIApplication)
instance applicationWillTerminate(_ application: UIApplication)
class
XCTestCase
inherits
class
BeatForFitTests
instance setUp()
instance tearDown()
instance testExample()
instance testPerformanceExample()
class
UITableViewController
struct
Constants
inherits
struct
Semaphore
instance let semaphore: DispatchSemaphore
instance wait(_ nanosecondTimeout: Int64) -> Bool
instance wait()
instance signal() -> Bool
instance init(value: Int = 0)
class
SettingsTableViewController
instance resetAllAlerts(sender: AnyObject)
instance viewDidLoad()
instance didReceiveMemoryWarning()
instance numberOfSections(in tableView: UITableView) -> Int
instance tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int


TempiFFT
TempiPeak
HeartRateManager
ProgressBar
TempiPeakDetector
Storage
BpmDetector
Player
NSObject
UITableViewDelegate
TableWithSongs
SmartSongSelector
Song
DataForTable
SongPicker
Pedometer
MPMediaPickerControllerDelegate
UITableViewDataSource
TempiBeatDetector
inherits
extends
conforms to



class
class
class
UITableViewCell
class
CellForLibrary
instance let var title: UILabel?
instance let var artist: UILabel?
instance let var albumImage: UIImageView?
instance let var tempo: UILabel?
instance set(_ song: Song)
extension
extension
instance viewDidLoad()
instance didReceiveMemoryWarning()
instance viewWillDisappear(_ animated: Bool)
instance presentPicker (_ sender:AnyObject)
instance loadList(_ notification: Notification)
class
LibraryViewController
instance let semaphore
instance let defaults
instance let var progress: UIProgressView!
instance var progressBar: ProgressBar?
instance let cellTableIdentifier
instance let storage
instance var songPicker : SongPicker?
instance let dataSource : DataForTable
instance let tableDelegate : TableWithSongs
instance let var table : UITableView! required
instance init?(coder aDecoder: NSCoder)
class
UIViewController
protocol
PedometerDelegate
instance updateCadence(cadenceIn: Int)
protocol
HeartRateDelegate
instance updateHeartRate(heartRate: Int)
extension
extension
instance showNoSongAlert()
instance notEnaughSongWarning()
instance playNextSong()
instance playPreviousSong()
instance setTitle()
class
PlaylistViewController
instance let storage
instance let player
instance let var table: UITableView!
instance let cellTableIdentifier
instance var nubOfRows
instance viewDidLoad()
instance didReceiveMemoryWarning()
instance viewDidAppear(_ animated: Bool)
instance tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
instance tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
instance tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Bool
instance tableView(_ tableView: UITableView, commit editingStyle: UITableViewCellEditingStyle, forRowAt indexPath: IndexPath)
instance playPlaylist(_ sender: AnyObject)
instance cleanPlaylist(_ sender: AnyObject)
extension
extension
instance updateHeartRate(heartRate: Int)
instance updateCadence(cadenceIn: Int)
class
PlayerViewController
instance let storage
instance let player
instance let smartPlayer
instance let heartRateMonitor
instance let pedometer
instance var isCadenceBlocked
instance var isHeartRateBlocked
instance let var segControl: UISegmentedControl!
instance let var playButton: UIButton!
instance let var songLabel: UILabel!
instance let var artistLable: UILabel!
instance let var bpmLabel: UILabel!
instance let var cadenceLabel : UILabel!
instance let var cadenceValueLabel: UILabel!
instance selectMode(_ sender: UISegmentedControl)
instance playButtonPushed(_ sender: AnyObject)
instance nextSong(_ sender: AnyObject)
instance previousSong(_ sender: AnyObject)
instance doSlow(_ sender: AnyObject)
instance doFast(_ sender: AnyObject)
extension
extension
instance var canBecomeFirstResponder : Bool
instance viewDidLoad()
instance viewWillAppear(_ animated: Bool)
instance remoteControlReceived(with event: UIEvent?)
inherits
inherits
inherits
inherits
conforms to
conforms to
extends

Bibliography

[i] Using music to tune the heart, Harvard Health Letter, November 2009 [Electronic resource]. URL: http://www.health.harvard.edu/newsletter_article /using-music-to-tune-the-heart (Date of access: 20.04.17)

[1] Hasso-Plattner Institute of Design at Stanford (d.school). An Introduction to Design Thinking PROCESS GUIDE [Electronic resource]. URL: https://dschool.stanford.edu (date of access: 15.09.2016).

[iii] Enhancing wellbeing: An emerging model of the adaptive functions of music listening / Groarke, J. M., Hogan, M. J. // Psychology of Music. 2015. 44(4), 769–791. doi:10.1177/0305735615591844

[iv] Wright, P. and McCarthy, J. Empathy and experience in HCI Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, ACM, Florence, Italy, 2008, 637-646.

[v] Reynolds, Gretchen. "Phys Ed: Does Music Make You Exercise Harder?", The New York Times, August 25, 2010. [Electronic resource] URL: https://well.blogs.nytimes.com/2010/08/25/phys-ed-does-music-make-you-exercise-harder/ (date of access: 13.11.2016)

vi Staff. "Upbeat Music Boosts Exercise Intensity", Medical News Today, October 20, 2003 [Electronic resource]. URL: http://www.medicalnewstoday.com/releases/4517.php (date of access: 13.11.2016)

[vii] Fauber, John via Milwaukee Journal Sentinel. "Workout with a tempo: Choice of music can affect exercise intensity", The Free Lance–Star, November 9, 2003

[viii] Waterhouse J, Hudson P, Edwards B. "Effects of music tempo upon submaximal cycling performance."// Scand J Med Sci Sports. 2010 Aug / 20(4): 662-9.doi:10.1111/j.1600-0838.2009. 00948.x PMID 19793214

[ix] Tenenbaum, G., Lidor, R., Lavyan, N., Morrow, K., Tonnel, S., Gershgoren, A., Meis, J., and Johnson, M. "The effect of music type on running perseverance and coping with effort sensations", Psychology of Sport and Exercise Volume 5, Issue 2, April 2004, Pages 89-109. doi:10.1016/S1469-0292(02)00041-9

[x] Laukka, Petric, and Lina Quick. "Emotional and Motivational Uses of Music in Sports and Exercise: A Questionnaire Study among Athletes." Psychology of Music 41 2011: 198-215.

[xi] Gretchen Reynolds. "How Music Can Boost Our Workout." The New York Times. October 23, 2013.

xii STATDATA (2016) Население России на 2016 год. [Electronic resource] URL: http://www.statdata.ru/russia (date of access: 23 October 2016).

xiii Богданов, А. (2014) Россия вошла в число лидеров по продажам iPhone. [Electronic resource] URL: http://appleinsider.ru/iphone/rossiya-voshla-v-chislo-liderov-po-prodazham-iphone.html (Date of access: 23 October 2016).

xiv TechNavio (2015) Global Fitness APP Market. [Electronic resource] URL: http://www.researchandmarkets.com/reports/3623668/global-fitness-app-market (Date of access: 24 October 2016).

xv Bevan J, Dransfield R, Coupland-Smith H, Goymer J and Richards C – BTEC Level 3 National Business Student Book 1, Pearson, 2009, ISBN 9781846906343

xvi Ron Jeffries "Essential XP: Card, Conversation, Confirmation". August 30, 2001 [Electronic resource]. URL: http://ronjeffries.com/xprog/articles/expcardconversationconfirmation/ (date of access: 10.04.2017)

[xvii] Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016. / Egham, U.K., February 15, 2017 / [Electronic resources] URL: http://www.gartner.com/newsroom/id/3609817 (date of access: 19.04.2017)

[xviii] Gamma E., Helm R., Johnson R., Vlissides J.: "Design Patterns", page 128. Addison-Wesley, 1995

xix Gamma E., Helm R., Johnson R., Vlissides J.: Design patterns: elements of reusable object-oriented software (14. print. ed.). Reading, Mass.: Addison-Wesley. p. 20. ISBN 0-201-63361-2.

xx Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; and Vetterling, W. T. "Correlation and Autocorrelation Using the FFT." §13.2 in Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed. Cambridge, England: Cambridge University Press, pp. 538-539, 1992.

[xxi] Lawrence Rosen, Open Source Licensing, p.85 Prentice Hall PTR, 1st ed. 2004.

[xxii] iOS Human Interface Guidelines [Electronic resource] URL: https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/ (date of access: 20.04.17)

[xxiii] Framework Core Motion by Apple Inc. [Electronic resources] URL: https://developer.apple.com/reference/coremotion (date of access: 17.04.2017)

xxiv Framework HealthKit by Apple Inc. [Electronic resource] URL: https://developer.apple.com/healthkit/ (date of access: 17.04.2017)

xxv A Guide to User Onboarding Techniques for Mobile Apps , Hannah Alvarez, November 11, 2015 [Electronic resource] URL: https://www.usertesting.com/blog/2015/11/11/user-onboarding-techniques-for-mobile-apps/ (Date of access: 23.04.17)

[xxvi] Material Design [Electronic resource] https://material.io (Date of access: 24.04.17)

xxvii The Amazing Audio Engine. SDK for processing audio on iOS [Electronic resource] https://github.com/TheAmazingAudioEngine/TheAmazingAudioEngine (Date of access: 13.08.16)

xxviii The Superpowered Audio SDK. [Electronic resource] http://superpowered.com (Date of access: 23.09.16)