Jog DJ Android App

CI360 – Mobile Application Development



Alena Ruprecht

Student No. 09800502

Submission Date: 24 May 2012

Table of Contents

1 Rationale of Jog DJ 3

1.1 Purpose 3

1.2 Functionality 3

1.3 Scenario of use 4

2 Technical Architecture 5

3 Development Process 9

4 User Research 10

5 Technical Difficulties & Solutions 11

6 Open Source Libraries 12

7 Critical Review 13

7.1 Strengths 13

7.1.1 Technically Ambitious 13

7.1.2 Good Software Development Practice 13

7.1.3 Polished Product 14

7.2 Weaknesses 14

7.2.1 Missing Pedometer Feature 14

7.2.2 BPM Is Calculated Remotely 14

7.2.3 Outdoor Testing Impedes Development 14

7.3 Future Improvements 15

Appendix A – Wireframe/Mock-up 16

Appendix B – System Architecture Overview 17

Appendix C – Project Planning Documents 18

Appendix D – User Research 21

# Rationale of Jog DJ

The target audience is people that run or walk for exercise. Currently there are no Android apps available that play a song matching a runner’s pace automatically. This is surprising considering Android’s market share of 47%.

Runners have to manually choose music that matches their speed. This can be tiresome and disturbs the flow of running. To have music that matches pace, a runner has to stop and pick a new song manually if he/she slows down or speeds up. This is of importance for runners that are in training. Training usually consists of three phases, a warm-up (slower speed), the actual run (faster speed) and a cool-down (slower speed). That means the user has to manually set a playlist to contain 3 different sets of songs in the right order for the right length of their training session. While this might be possible for one run, the same playlist becomes boring for multiple runs.

A training plan and audio-cue feature help by announcing each phase a runner is in. This will save the runner having to look at the time on the phone and thus avoiding accidents by not paying attention.

## Purpose

The purpose of Jog DJ is to facilitate a runner’s life by automatically playing songs that match the runner’s pace and keep the runner updated about running statistics like pace, distance, time and training phase.

## Functionality

**Playlist feature:** The main feature is to automatically create a playlist based on pace.

**GPS feature:** The app allows the user to track running statistics like pace, distance and time passed. The GPS feature visually draws the runner’s route on a Google map.

The feature brings the runner from the start screen to a new screen displaying a map and giving access to music controls underneath allowing pausing, forwarding or rewinding the music.

**Audio cue & training plan feature:** There a two kinds of audio cues. If a time or distance limit is set in the training plan feature, the audio cue will automatically announce each phase and running statistics. If no training plan is chosen, the runner can still hear audio cues of running data by accessing the settings screen and selecting that option.

## Scenario of use

Kara opens the app and selects the training plan icon. She sets her warm-up to 5 minutes, the main run to 5 kilometres and the cool-down to 5 minutes. Back on the main screen, she pushes the start button, puts her phone into a running armband and starts running.

An audio cue tells her to start the warm-up phase and after 5 minutes to start the main run for 5 kilometres. During the run Kara forwards a song she doesn’t like. After 5 km, the audio cue announces the cool-down phase.

When the run finishes, Kara looks at the screen to view her route on the map.

Further scenarios can be found in Appendix D.

# Technical Architecture

The diagram in Appendix B shows the architecture of the application. This section will explain how these classes are interacting with each other by feature.

When the application starts the first activity is the *JogDJActivity* presenting the user with the main screen. This activity is responsible for starting the four services essential to the application.

**Music feature:** The *MusicFinderService* is responsible for finding all the songs available on a user’s device and getting the BPM (or tempo). Scanning the device for songs is done through the Android API and turned out to be easy. However, calculating the BPM of each song proved to be a bit more complex. The initial plan was to use a third-party library and perform the calculation on the device. This solution did not work because no Java library working on Android could be found. The alternative solution chosen was to use a web based service called EchoNest that makes it easy to retrieve the BPM of a song knowing its title and artist. EchoNest is simply a huge database of songs that can be queried using an API. Once the BPM of a song is found all its information is stored in the application database so that the above process only has to be done once. Whenever a song is added to the database, the *MusicFinderService* broadcasts a message to notify other components that new songs are available.

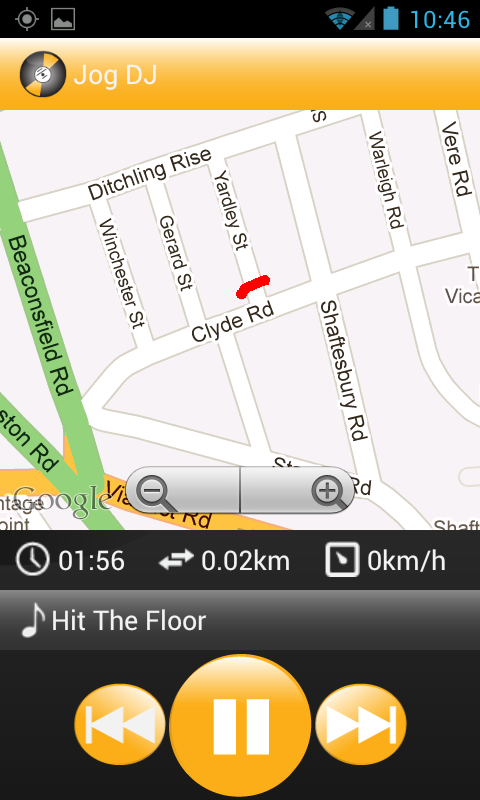
Most of the operations described above can be time consuming. They are therefore performed on a background thread in order to avoid blocking the UI thread.



Figure 1 - Song Explorer displaying all available songs

**GPS feature:** The user sees the RunMapActivity once a run has been started. It displays information about the run such as current pace, distance and the chronometer. The activity is also responsible for playing songs based on current pace. It does not do a lot of work and relies on the *RunMonitorService* to get information about the run.

*The RunMonitorService* is responsible for monitoring the current location, which is used to calculate the speed and the distance. The current location is obtained from another service called the *LocationService*. This service is simply a wrapper for the Android location manager. Whenever a new GPS reading is available the *LocationService* broadcasts a message that notifies the *RunMonitorService* about the new location. The *RunMonitorService* then calculates the pace and distance based on this new location. Finally, it broadcasts a message to notify the rest of the system that new information about the run is available. The *RunMonitorActivity* receives this message and displays the new pace and distance under the map. It also uses the new location to plot the user’s position on the map. Finally the pace is used to determine which song should be playing. The *MusicChooser* class matches the speed to BPM and finds the most appropriate song in the application database. Once a song has been found the *MusicPlayerService* plays it.



60

Figure 2 - Map with music controls, running statistics, song & BPM and runners path plotted

**Audio cue & training plan feature:**

This screen offers the user to input time or distance for running phases. For example the user can choose the warm up to last 10 minutes, the main run 8 kilometres and the cool down 5 minutes. At the beginning of a phase the application plays an audio message informing the user about the length of the coming phase.

The implementation is fairly straightforward. The *RunPlanningActivity* lets the user select the audio cues he wishes to receive. His preferences are saved in the application database and then used by the *RunMapActivity* at the beginning of the run. The activity creates a *TimerTask* that verifies every second whether the condition for playing an audio cue is met - when the runner has ran for 10 minutes. When this condition is met the application moves to the next phase and plays an audio message using Android’s text to speech feature. This feature is extremely easy to use. After initialising it, it is just a matter of calling a function and passing it a string containing the message to play. The audio cue doesn’t have to be pre-recorded and can be modified depending on context.

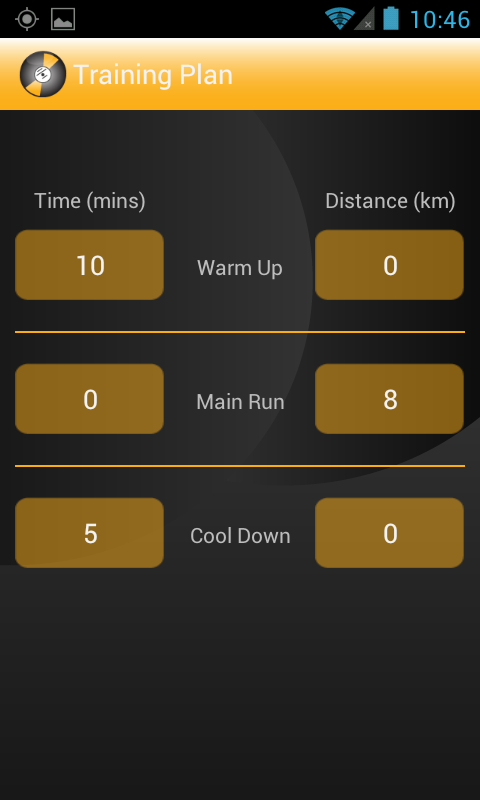


Figure 3 - Training Plan Screen

**Settings screen:** This screen offers runners to pick the unit of length the application should use. This screen also lets the user set audio cues based on time or distance. For example selecting “3 minutes” means that the application will play an audio message every 3 minutes. This message sums up the running data. Similarly to the training plan this feature is implemented by creating a *TimerTask* that checks every minute whether a trigger condition is met. It plays an audio message using the text to speech functionality.

The developer mode can be selected for debugging purposes. It rewires the music controls in the *RunMapActivity* to manually adjust the speed. In this mode the next button increases the speed by 0.1 and the previous button decreases it by 0.1. This makes it possible to test how the music adjusts when the speed changes.

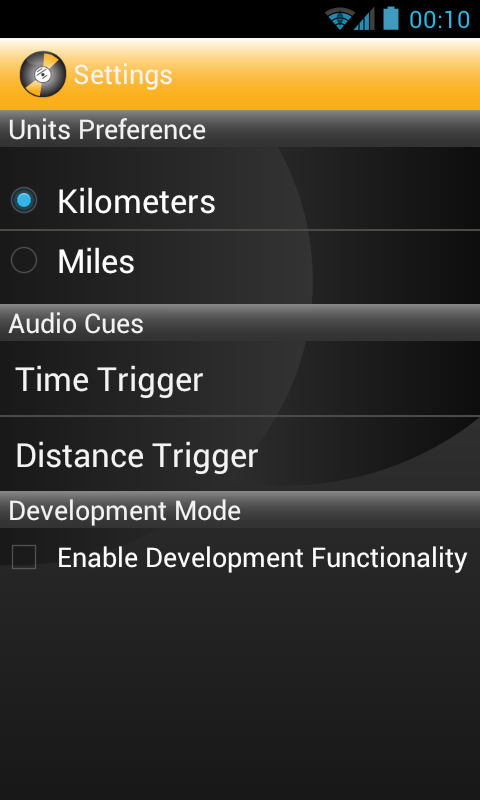
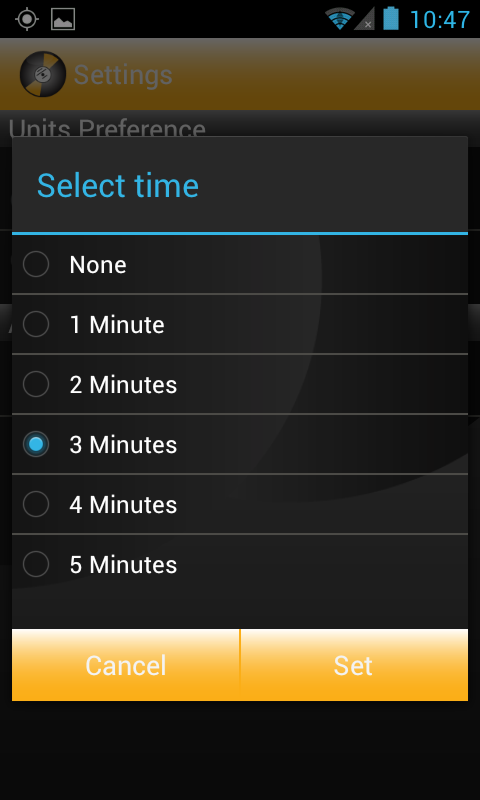
 

Figure 4 - Settings Screen and open dialog window

# Development Process

The development followed an iterative process as proposed in the project proposal. After careful planning and collecting requirements from potential users as well as research based on existing running and music apps, mockups were drawn up and tested.

Testing with personas identified various usability issues, which triggered a review of the layout and navigation. A new mockup was drawn, tested and the actual development process started.

The iterative process allowed for constant changing of requirements and design, which influenced the development process. The features of the application allowed the development to be divided into modules for development and can be found in Appendix C.

Each module was developed and field-tested. During every evaluation it was discovered what worked and what didn’t. Problems were then addressed in the next round of development. As this was a small project with one developer, the iterative process was very fast and allowed addressing issues quickly.



The timeline of development was closely followed apart from switching the pedometer to be the last feature. During research it was discovered that it would be easier to develop the GPS feature first. It seemed sensible to develop all features to make the app work first and if time allows add additional features. The pedometer feature has not been implemented in the end due to time constraints. Working out the core structure of the app turned out to be fairly straightforward, however a lot of time was spent in polishing details like user preferences, dialogs and fine-tuning the algorithms.

# User Research

Users interested in the application will be primarily runners that listen to music while they train. The idea of the app was perceived during my marathon training. Not having enough time to create playlists for each run, I listened to the music randomly played by my phone. However, this became quite irritating due to the fact that music would play with different speeds. This meant that when running fast and a slow song came on, I had to stop, unlock the phone and change the song or do all this while running not paying attention to the environment.

I undertook a persona-based expert review on the mock-up of the application. This research was rather extensive due to the lucky fact that I am also taking the Usability Evaluation module and evaluated the app based on heuristic principles.

The research process is detailed in Appendix D along with the results.

# Technical Difficulties & Solutions

The core structure of the app was fairly straightforward to develop. Problems were encountered when fine-tuning the features.

**Problem 1**

Working out which song should play was a fine-tuning process that took most of the time, as it is difficult to test a running app while sitting in front of a desk. As explained previously the application matches the current pace to BPM and making this work involves getting reliable speed information. This was a problem because the GPS doesn’t always give accurate information. This issue was solved by using the last ten GPS readings to calculate the current speed which gave much more reliable results. Calculating an average speed allows attenuating the effect of incorrect GPS readings. A better solution would probably be to discard GPS locations that are too inaccurate.

The next challenge was to devise an algorithm that would pick a song based on the current speed. Through a lot of experimentation the following algorithm was chosen. First the application decides whether the song currently playing should be changed. In order to avoid picking a new song every time the speed changes slightly, the algorithm checks that the speed has not changed by more than 10% in the last thirty seconds. When that is the case the code turns the speed into BPM and tries to find a song matching the value in the database. When a song is found the algorithm tries to find another ten songs that are within twenty BPM of the target BPM. Once this is done the algorithm picks a random song from the list of ten, or less, candidate songs. This process is done to introduce a bit of variety and avoid always playing the same song. Finally the algorithm checks that the song picked is not within twenty BPM of the song currently playing. In that case the new song is discarded and the current song keeps playing. It is better to let the current song playing instead of replacing it with a song with a similar BPM.

**Problem 2**

The original project proposal envisioned that the BPM calculation would be done on the device. This proved futile after researching various libraries without result and the third party service EchoNest was utilised instead. EchoNest offers a huge database of songs that can be queried. For each song the database stores information such as title, genre, artist, BPM, etc. Using EchoNest means that the user needs an Internet connection when using the app for the first time because the application needs to query the EchoNest database. Once song information has been gathered, it is stored in the application database for further use. One weakness with this solution is that EchoNest imposes a limit of 120 queries per minute. The *MusicFinderService* works around this limitation by ensuring that no more than 100 songs are processed every minute.

**Problem 3**

The Android music player is a state machine and before proceeding with any operation, it needs to be ensured that the player is in the correct state. Each time a song stops and starts again, the music player has to be reinitialised which in itself is not complicated but can be a source of errors if the documentation is not read carefully. Sometimes an mp3 song is queried and stored in the database but the Android player can’t play the content. No solution has been found to this problem, as it couldn’t be established why Android can’t play the song, probably a formatting issue.

**Problem 4**

Originally the app was set to target Gingerbread (Android 2.3.3). However the design of the Holo theme was very appealing and is only available with Ice Cream Sandwich. As the app doesn’t use any features from the 2.3.3 version and the older versions don’t support the ActionBar, the manifest now targets 4.0.3 and uses the Holo theme as parent theme in the customized theme.

**Problem 5**

Styling the app proved to be a very time consuming task. The bitmaps were created in Illustrator in various sizes, however moving the bitmaps to different resource folders actually results in them being scaled differently when on the same device. Once this peculiarity was noticed, all bitmaps were stored in the drawable-hdpi folder and smaller versions in the drawable-mdpi folder.

Another note on styling is that it sometimes felt that the styling was very CSS-like with assigning padding and margin, etc. An improvement might be to develop with templates instead.

# Open Source Libraries

The third-party service EchoNest[[1]](#footnote-1) is used to retrieve the BPM of the songs stored in a user’s music library.

The open source library JAudio[[2]](#footnote-2) is a Digital Signal Processing project built to provide an easy-to-use program for audio feature extraction. It was used to extract the metadata from the mp3’s.

Advice and help was found and searched for on www.stackoverflow.com as well as the Android documentation.

# Critical Review

This section discusses strengths and weaknesses of the project, the development process and suggests future improvements. Overall I consider this project a success in terms of outcome, process and knowledge gained. After addressing the weaknesses discussed below, the app will be published on the Play store (former Market).

## Strengths

### Technically Ambitious

The ambitious design made it necessary to overcome various technical challenges.

**Detecting Music:** The app manages to detect a user’s library and sends off the metadata to a third party service in order to calculate the BPM. This solution is quite elegant because it relies on the cloud to do the heavy work and results are available almost immediately. Calculating the BPM on the device would require a lot more time and would also use a lot more battery.

**Matching Music:** The algorithm behind matching the music to the speed was heavily tested and results in a pleasant experience for the user. The current speed is averaged overtime to attenuate the effect of GPS inaccuracy. The algorithm choosing the next song was tweaked to provide a smooth experience. It ensures that the speed is consistent for a period of thirty seconds before changing songs to avoid a “zapping” music experience. The algorithm also introduces variety by not always picking the same song.

**Audio Cues**: The running experience is enhanced by adding audio cues to the runner’s routine allowing to solely focus on the run. The runner doesn’t have to interrupt the run to check the clock or to change songs. The implementation is flexible enough to work for runners with a training plan as well as runners that just want audio cues about time and distance giving the app a well-rounded feel.

### Good Software Development Practice

The developer uses good software practice to develop this application.

The various features of the application are encapsulated in packages. Public attributes are avoided as they make the inner working of a class accessible to the rest of the application. Getters and/or setters are implemented where necessary. This level of abstraction makes the code easier to use, maintain and extend. For example the *RunMonitorService* does not know how the speed is calculated. All it knows is that another service regularly sends a message with the newly calculated speed. This architecture means that it would be very easy to use a pedometer instead of GPS to calculate the speed. Only the service calculating the speed would need to change.

### Polished Product

The app contains many features that give it the feel of a commercial app. It is professional in the light of its production and content. It contains various screens including a settings screen and a help screen if any questions arise. The app is designed to address the simple needs of runners and incorporates user research to ensure that the app facilitates a runner’s life with easy controls instead of adding complications. It also includes a developer setting allowing testing the app without leaving the desk.

It is not a commercial product yet but it is already fully featured.

## Weaknesses

### Missing Pedometer Feature

The app was supposed to work independent of network connections by using a pedometer feature. Even though it was possible to find open-source code offering fully functional pedometers, they were not exactly what was needed and would have to be slightly rewritten. It would have been a time consuming feature to implement and in the end there was not enough time. This means the app has to rely on GPS data, which might be a problem for some users but on the other hand most smartphones offer this nowadays.

### BPM Is Calculated Remotely

A further change of the original requirements is the calculation of BPM remotely. Users have to have Internet connection to query the EchoNest database. To mitigate this issue the application stores song information in a database. The Internet connection is only needed the first time a song is added.

Another downside of using EchoNest is that the app relies on another service that can stop responding, provide inaccurate data or not know the song that is requested making it more difficult to provide a consistent user experience.

### Outdoor Testing Impedes Development

As the app is for outdoor use mainly using GPS data, it was quite difficult to check the validity of the algorithm choosing the song to play. GPS data is not 100% accurate causing big jumps in pace, which in turn causes the music to change inappropriately. Even though a lot of care was taken to determine the accurate pace, the formula might not be perfect.

## Future Improvements

The app’s requirements changed during the development process due to time constraints and difficulty testing outdoors. At the moment the user only gets one input option to use the app. This could be improved by adding the pedometer function making the app available offline (once the library is scanned).

The app is currently targeting 4.0.3 making it only accessible to 10% of the market. As it currently doesn’t use any major features of that target version apart from the theme and ActionBar, it should be made available for lower targets by manually adding the ActionBar with an additional TextView or using ActionBarSherlock[[3]](#footnote-3) to make it commercially viable.

TextView

        android:layout\_width="fill\_parent"

        android:layout\_height="50dp"

        android:background="@drawable/actionbar"

        android:text="@string/app\_name"

        android:drawableLeft="@drawable/icon"

        android:padding="5dp"

        android:gravity="center\_vertical"

        android:textAppearance="?android:attr/textAppearanceLarge"

android:drawablePadding="10dp"/>

Another feature that the app could offer is the exclusion of specific genres in the Settings screen allowing the user to exclude genres they don’t want including audiobooks.

Saving the runner’s path and statistics in a database could be another addition connecting with social networks like Facebook or Twitter allowing the runner to share their run data with friends.

Before making the app commercially available it should run through an extensive testing procedure to ensure that the music-matching algorithm will function under any circumstance.

# Appendix A – Wireframe/Mock-up

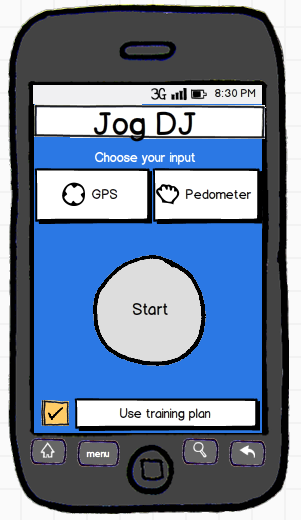
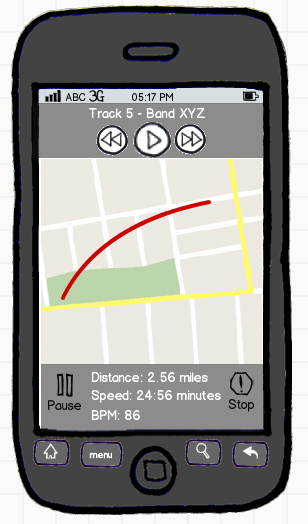
 ****

Figure 5 - Initial wireframes created in Balsamiq

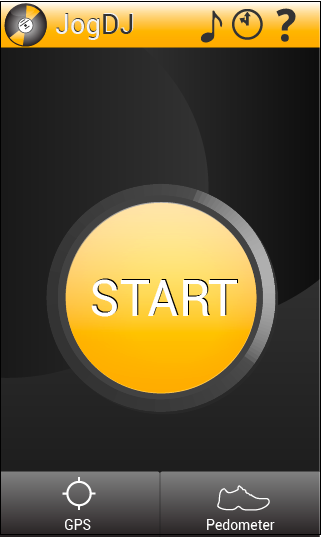
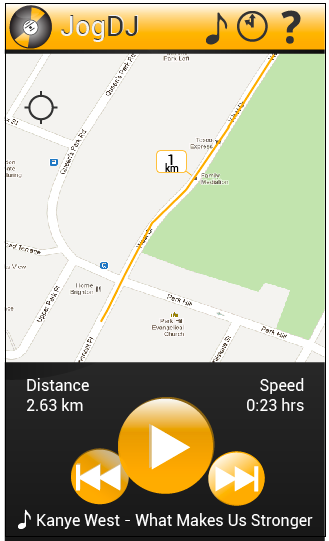
 

Figure 6 - Mockups created in Illustrator (parts also used for styling the app)

# Appendix B – System Architecture Overview

Plays song

Android

Manifest &

main.xml

Data

base

Checking for next song

Plays song

Function call

Reads

from

Writes to

Sends Message

# Appendix C – Project Planning Documents

(The following documents are from the original project proposal.)

The developer is proposing an iterative approach to this project due to the nature of the project having to be tested in a real-life environment.

Initial Planning

Deployment

The initial planning phase has been done with this proposal. Extensive research into the separate components of this application has also been undertaken.

The project has been divided into several sub tasks and will be developed accordingly. The iterative development plan will approach each of the following phases:

* Access to the GPS system
* Calculating distance by time with the GPS
* Displaying GPS output on Google Maps
* Access to the accelerometer and directional compass to output motion into footsteps
* Access to the built-in music library
* Calculating bpm for a song
* Matching bpm to running speed
* Accessing a song and playing it back
* Registering new input on speed and change of song
* Voice-over features for the training plan

Once a planning phase has been concluded, requirements will be set for each sub task. The design and analysis will only approach the sub task and this sub task only. Once this has been done satisfactorily, the task will be implemented and tested. As the GPS and accelerometer are the first tasks to be implemented, they will have to be tested outdoors to determine if they correctly in- and output the information.

The test results have to be evaluated and if satisfactory, the next sub task can be started.

The first sub task is the implementation of the GPS positioning and it’s output to a Google Map. The task can take up to two weeks, however due to the difficulty of this task, it is anticipated that it might take longer to complete thus taking us into the timeframe of the next sub task. Should this happen, the implementation of the accelerometer feature could be omitted. The application can function solely with GPS and the accelerometer is just an extra feature.

Each evaluation will add to the write-up of the project. This is to ensure consistency across all stages rather than having to write up from memory at the end of the project. There will be time dedicated to the task of polishing the report and adding a conclusion.

The proposed timeline is shown in the below diagram:



Each of the above stages includes all phases of

* Planning
* Requirements gathering
* Analysis
* Design
* Implementation
* Testing
* Evaluation
* Additional iterations if needed

at the end of each task. This is to ensure consistency and help adding features only when the current stage has successfully been completed.

The third last milestone on the timeline is dedicated to ensuring that the delivered product is at the highest possible standard by eradicating bugs through extensive testing in the field.

The second last milestone is to ensure that the project is delivered with a report that rivals equally in accuracy and detail. The last milestone is considered to be the actual hand-in on May 24th, 2012.

# Appendix D – User Research

**Abstract Of Results**

The application interface is very simple and consists of buttons that guide the user through the application. The big yellow start button invites the user to just start the app, however the icons in the tab bar below allow the user to make further selections based on personal preferences.

Design decisions such as font size, colour choices, etc. were made based on feedback from the assumed persona and the app navigation was restructured.

The overall structure, navigation and design of the app are suitable for the user’s task and require very few skills in starting the app. The user can focus on the using the app rather than having to figure out how the app works, as the procedure is clear through inviting imagery and text.

The app behaves according to the users expectation and conforms to user expectations across all screens. Due to the simple design, it is easy to remember how to use the app thus enforcing learnability.

The app can be retracted at any point allowing the user to change screen and sequence of interaction because of a change of heart or any errors.

There are currently no functionalities that address accessibility apart from changing the preference of units used system-wide.

**Method**

This section discusses the applied method for mobile usability testing and the steps involved to create the evaluation.

Personas were created to use as guidance for task scenarios. To create personas, I imagined the kind of people that would use the Jog DJ app and came up with two basic distinctions between professional runners, who would use the app during training as well as amateur runners. The app has a very specific purpose, thus the scenarios were very similar with slight differences in setup.

The evaluator was briefed on the heuristics of the *Ergonomics of Human System Interaction* standard and evaluated each task from the perspective of a persona with heuristics in mind.

[**BS EN-ISO 9241-110**](http://www.userfocus.co.uk/resources/iso9241/part110.html) **- Ergonomics of Human System Interaction**

**1) Is the dialogue suitable for the user's task and skill level? (Suitability for the task)**

“A dialogue is suitable for a task when it supports the user in the effective and efficient completion of the task. In a dialogue which is suitable for the task, the user is enabled to focus on the task itself rather than the technology chosen to perform that task.”

**2) Does the dialogue make it clear what the user should do next? (Self-descriptiveness)**

“A dialogue is self-descriptive to the extent that at any time it is obvious to the users which dialogue they are in, where they are within the dialogue, which actions can be taken and how they can be performed.”

**3) Is the dialogue consistent? (Conformity with user expectations)**

“A dialogue conforms with user expectations if it corresponds to predictable contextual needs of the user and to commonly accepted conventions.”

**4) Does the dialogue support learning? (Suitability for learning)**

“A dialogue is suitable for learning when it supports and guides the user in learning to use the system.”

**5) Can the user control the pace and sequence of the interaction? (Controllability)**

“A dialogue is controllable when the user is able to initiate and control the direction and pace of the interaction until the point at which the goal has been met.”

**6) Is the dialogue forgiving? (Error tolerance)**

“A dialogue is error-tolerant if, despite evident errors in input, the intended result may be achieved with either no or minimal corrective action by the user. Error tolerance is achieved by means of damage control, error correction, or error management to cope with errors that occur.”

**7) Can the dialogue be customised to suit the user? (Suitability for individualisation)**

“A dialogue is capable of individualization when users can modify interaction and presentation of information to suit their individual capabilities and needs.”

**Personas**

|  |  |  |
| --- | --- | --- |
| **Amateur training for the marathon** | | |
| Kara G |  |  |
| Macintosh HD:Users:alena:Downloads:kara.jpg | Kara lives just outside the city centre of Brighton with a couple of housemates.  She works at a software company as a receptionist and enjoys the balance of an office job and her fitness regime of running, swimming and Zumba.  *Kara is turning 30 soon and wants to run the marathon before. She enjoys listening to music while training but this is her first marathon and sometimes the training is very hard.* | 1. 29 years old 2. Lives with her housemates Emma and Julie 3. Has a boyfriend who does not like running 4. Works as an office receptionist for a software company 5. Only started training for the marathon 4 months ago   **What are your interests?**  *“Zumba is my favourite exercise but since I started running, I love that too. And I have a feeling that I’ll continue with it even after the marathon.”* |
| Kara’s Goals | * Enjoy music without having to manually create a play list * Knowing her speed | |

|  |  |  |
| --- | --- | --- |
| **Professional free runner** | | |
| Jonathan M |  |  |
| Macintosh HD:Users:alena:Downloads:jonM.JPG | Jon lives in Lewes but is frequently traveling to compete in free running tournaments.  He lives with his wife Sara and their 2-year-old daughter Lucy.  In his spare time, he loves working around the house and spending time with his family.  *Jon made his hobby his profession and loves listening to music while running around and over things.* | 1. 34 years old 2. Lives with his wife Sara 3. Has a 2-year-old daughter Lucy 4. His work often takes him away from his family and he misses them   **What are your interests?**  *“Free running is not just a sport, it’s a passion. I’m really lucky that I can run for work and still love doing it in my free time.”* |
| Jon’s Goals | 1. Knowing his route 2. Listening to music while he’s training 3. Having a training plan | |

**Task Scenarios**

|  |  |  |
| --- | --- | --- |
| **What is it?** | **Case Study 1** | **Case Study 2** |
|  | Jonathan M | Kara G |
| Scenario | Using the pedometer feature to run. | Using the GPS feature with the training plan to run. |
| Practical Goal | Jon wants to measure her running speed and listen to music. | Kara wants to know how far she has to run using the GPS feature and makes use of the training plan feature timing her warm-up, main run and cool-down. |
| Time | It’s 10am and Jon is going for a free run through the forest to keep himself fit. | It’s after work and Kara wants to go for a training run after work. She doesn’t want to keep checking her watch, so she relies on the app to tell her when it’s time to speed up. |
|  | 1. Jon opens the app. 2. He selects the pedometer feature. 3. He clicks the start button. (The music starts playing). 4. He puts the phone into his armband. 5. He starts running. 6. After the run, he stops the app. | 1. Kara opens the app. 2. She selects the GPS feature. 3. She selects the Training Plan feature. 4. She selects the warm-up dropdown and sets it to 15 minutes. 5. She selects the main run dropdown and sets it to 2 hours.  6. She selects the cool down dropdown and sets it to 10 minutes. 7. She uses the back button and returns to the previous screen, the Start Screen. 8. She clicks the start button. (The music starts playing). 9. She puts the phone into an inside pocket of her running shorts. 10. She starts running. 11. A voice-over informs her of the run phases she is in. 12. She finishes her run after 2 hours and 30 minutes. 13. She stops the app. |
| Environment | Near Devil’s Dyke, a national reserve (energized) | Outside Kara’s home (energized) |
| User | Professional runner (Jonathan M) | Leisure runner (Kara G) |

**Process Steps for Persona-Based Expert Review**

Here is a short overview of the activities that were completed for the persona-based expert review.

**Research**: A short study into runner’s behaviour, professional and amateur alike, during training has been undertaken. The behaviour was noted and the app has been designed around this observation. Runners were interviewed about their running preferences and observed when running with a music app or a running app. The conducted interviews asked the runners about their likes and dislikes about running with the existing apps and these were taken into account when designing the UI of the Jog DJ app.

**Modelling**: A set of personas was created based on the research. Each persona’s expectation and mental model were determined. A set of high-level abstract context scenarios for each persona was developed (see Appendix B).

**Evaluation**: The evaluation determines where the application provides good support for the scenarios and where problems are observed. These are noted and assessed in the next stage. As this project only comprises the pilot test, the results are preliminary and have to be verified in further evaluations. The pilot tester executed each persona’s context scenario based on the set of principles in Appendix A.

**Reporting**: Preliminary results for the pilot test have been compiled for this paper and a report would have to be further elaborated after a full evaluation. Typically, problems would be correlated with principles at this stage, described and prioritized.

**Test Procedure**

The mock-up consisted of multiple screens. A mobile screen with the Jog DJ icon was presented. When the icon was pressed, the app “opened” and the screen was replaced with the Start Screen. Using the perspective of a user, the Start Screen was evaluated for colour scheme, branding and positioning of elements. The evaluator would evaluate the first screen based on the heuristic principles as described above and note them.

The evaluator was then presented with the first task. She talked aloud about what she was doing if she was about to go running. The test results can be found below. The evaluator tapped on the screen indicating what the persona would do and the screen was replaced depending on the selection.

Depending on the button that was pressed, a new screen or a response was presented. Each thought process was compared to one of the heuristics and how these were fulfilled. This was repeated for each task and each persona.

After the evaluation of each set task for the first persona, the evaluator “stopped” being a user and evaluated the overall application based on heuristics. This procedure was repeated for the second persona and differences in the use of the application noted. Both personas have slightly different needs and it was endeavoured to get an understanding of how a user would interact with the application from different perspectives.

**Results**

Usability comments were classified in one of the following categories:

|  |  |  |
| --- | --- | --- |
| Category code | Status | Details |
| Category code C | Positive finding | This approach is recommendable and should be preserved. |
| Category code P | Minor problem | May cause customers to hesitate for a few seconds. |
| Category code Q | Serious problem | May delay a considerable number of users in their use of the application for 1 to 5 minutes, but eventually they will be able to continue. May cause occasional “catastrophes”. |
| Category code R | Critical problem | May cause frequent catastrophes. A catastrophe is a situation where the application “wins” over the test participant, i.e. a situation where the potential user may not be able solve a reasonable task or where the application annoys the customer considerably. |
| Category code  T | Bug | The application works in a way that’s clearly not in accordance with the design specification. This includes spelling errors, dead links, scripting errors, etc. |

Preliminary pilot test results:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Category | UI Area | Description |
| 1 | P – Minor problem | Overall | **Brand identity is missing across screens.**  There is no identity of the brand on the consecutive screens. This gives the user the impression that the screens may not belong together. |
| 2 | P – Minor problem | Overall | **Colour choice is inconsistent across screens.**  The colour choice is very bland and the colour is inconsistent across the different screens. |
| 3 | C – Positive finding | Overall | **Simple navigation is appealing.**  The application is presented in a minimalistic style offering mainly to play music without any difficult decision-making. |
| 4 | C – Positive finding | Overall | **Screens are free from clutter.**  The screens are very designed to be straightforward and have a very nice button design that allows the user to quickly grasp the application. |
| **Start Screen** | | | |
| 5 | Q – Serious problem | Start Screen | **Order of workflow is confusing.**  The big start button invites to be clicked first. This is misleading, as the GPS/Pedometer input has to be chosen first.  *Suggested solution*: Grey out the areas that can’t be used until the user has chosen an input method or offer a default method. |
| 6 | Q – Serious problem | Start Screen | **The check box next to the button is confusing.**  The check box next to the training plan button is not unchecked by default. This is confusing for the user as the training plan has no default options and when the user clicks start a pop-up comes up warning the user that training options have to be selected.  *Suggested solution:* Uncheck the training plan by default and when ticked, automatically redirect to the screen. Alternatively remove the check box and leave the training plan button on it’s own but set the training plan to not chosen by default and only redirect if the user clicks the button. |
| **Map Screen** | | | |
| 7 | C – Positive finding | Map Screen | **The music controls are straightforward to use.**  The controls are easy to understand and users were familiar with them due to the similarity to other music applications. |
| 8 | P – Minor problem | Map Screen | **The map is too simple and not detailed enough to indicate what it does.**  It’s not immediately clear what the user can do with the map like zooming, how long the run was, etc.  *Suggested solution:* Display icons that indicate zooming like plus/minus symbols and mile markers to indicate the running distance. This would allow the runner to keep track of the run when looking on the screen. |
| **Map Screen / Pedometer Screen** | | | |
| 9 | Q – Serious problem | Map Screen / Pedometer Screen | **Font size is very small on the screens.**  The playback information gives sufficient information about the music and the runner’s distance/speed. It’s quite small and might be difficult to read on a small device.  *Suggested solution:* Offer a settings menu that allows the user to pick preferred font size and contrast colours. |
| 10 | P – Minor problem | Map Screen / Pedometer Screen | **Additional training information not needed.**  The training plan information is very small and strongly contrasted. As the training plan is a voice-over, the information might not be needed on first glance.  *Suggested solution:* Hide the training plan information in a button that displays information when pressed in a bigger font. |
| 11 | R – Critical problem | Map Screen / Pedometer Screen | **Two pause buttons and a stop button are misleading.**  It’s confusing that there are two pause buttons. One at the top and one at the bottom. One gives the appearance of stopping the music, the other pauses the app. The stop button does the same thing.  *Suggested solution:* As the music is the main part of the app, it might be better to remove the bottom pause & stop button and move the stop button up to the other controls. That way there is a consistent area with controls and only the feedback information is displayed at the bottom of the screen. |
| 12 | Q – Serious problem | Map Screen / Pedometer Screen | **No choice of excluding music.**  The user can forward an unwanted track but has no input about which music from the library is played.  *Suggested solution*: Offer the user the option to exclude specific genres/tracks/artists in a settings screen. |
| **Pedometer Screen** | | | |
| 13 | P – Minor problem | Pedometer Screen | **BPM display seems “empty”.**  The pedometer screen feels “empty”. It only shows the BPM in the centre of the screen. This might be ok when running but doesn’t indicate the fact that music is playing.  *Suggested solution*: Instead or in addition to the BPM, display an indicator that music is playing. This could be a moving graph or a circle indicating that a song is playing like below:  83  Macintosh HD:Users:alena:Desktop:Screen Shot 2012-04-24 at 13.24.55.png |
| 14 | Q – Serious problem | Pedometer Screen | **Missing distance display.**  The pedometer screen doesn’t display the running distance, only the time.  *Suggested solution*: Display the running distance like on the map screen (for familiarity). |
| **Training Plan Screen & Voice Over** | | | |
| 15 | C – Positive finding | Training Plan Voice Over | **Voice-over lowers music volume.**  When the voice over is talking, it lowers the music volume simultaneously. This feature is very nice and allows the runner to interact with the app “hands-free”. |
| 16 | Q – Serious problem | Training Plan Voice Over | **No choice of voice or volume.**  The voice over is automatically set when the training plan is selected. It is not possible to choose between different voices or different volumes.  *Suggested solution:* Offer a settings menu that allows the user to set preferences on voice and volume. |
| 17 | Q – Serious problem | Training Plan Screen | **Confusing dropdown selection.**  The screen is very simple. It offers dropdown boxes for the areas user want to include in their training. Very straightforward but maybe a bit confusing that the non-selected dropdown boxes are still visible.  *Suggested solution:* Grey out the areas that aren’t selected to make it more obvious that these areas are not working. |
| 18 | Q – Serious problem | Training Plan Screen | **Dropdown boxes only offers time selection.**  The dropdown boxes are very simple but only offer the user to choose between different times. Runners also have to train based on distances.  *Suggested solution:* Offer the user the option to choose between time or distance dropdown boxes or a mix of both. |
| 19 | Q – Serious problem | Training Plan Screen | **Explanation of what the training plan screen does.**  It is not immediately clear what the dropdown boxes signify even though the times are displayed.  *Suggested solution*: Display an instruction above the dropdown boxes to explain. |
| 20 | P – Minor problem | Training Plan Screen | **Total running time on training plan.**  The training time is displayed in three segments and not together.  *Suggested solution*: There should be a total at the bottom of the screen showing the users selections. |
| 21 | P – Minor problem | Training Plan Screen | **Redundant back button.**  There is a back button on the top of the screen. It is redundant as there is a built-in back button on Android phones.  *Suggested solution*: Remove the back button. |

1. EchoNest at: http://developer.echonest.com/index.html [↑](#footnote-ref-1)
2. JAudio at: http://jaudio.sourceforge.net/ [↑](#footnote-ref-2)
3. ActionBarSherlock is an extension of the [compatibility library](http://developer.android.com/sdk/compatibility-library.html) designed to facilitate the use of the action bar design pattern across all versions of Android with a single API. Available at http://actionbarsherlock.com/ [↑](#footnote-ref-3)