****

**Team 52: Jaromir Latal & Lambros Zannettos**

**Client: The Mind Manifesto**

**(Imran Ahmad, Terry Hall)**

**Supervisor: Dr. Yun Fu**

****

**Table of Contents**

**1.** [**Table of Contents** 3](#_Toc447897868)

[**2. List of Figures** 4](#_Toc447897869)

[**3. Abstract** 5](#_Toc447897870)

[**4. Context** 5](#_Toc447897871)

[**4.1. Project Background** 5](#_Toc447897872)

[**4.2. Problem being solved** 5](#_Toc447897873)

[**5. Team Roles and Skills** 6](#_Toc447897874)

[**5.1. Jaromir Latal** 6](#_Toc447897875)

[**5.2. Lambros Zannettos** 6](#_Toc447897876)

[**6. Requirements** 7](#_Toc447897877)

[**6.1. Initial questions** 7](#_Toc447897878)

[**6.2. List of requirements** 7](#_Toc447897879)

[**6.3. Design** 8](#_Toc447897880)

[**6.4. Use Cases** 10](#_Toc447897881)

[**7. Work Distribution** 12](#_Toc447897882)

[**7.1. Jaromir Latal** 12](#_Toc447897883)

[**7.2. Lambros Zannettos** 12](#_Toc447897884)

[**7.3. Shared** 12](#_Toc447897885)

[**8. Technical Design** 13](#_Toc447897886)

[**8.1. Technology chosen** 13](#_Toc447897887)

[**8.2. Structure of the Application** 14](#_Toc447897888)

[**8.3. User Interface** 14](#_Toc447897889)

[**8.4. Design Patterns** 16](#_Toc447897890)

[**8.4.1. Singleton** 16](#_Toc447897891)

[**8.4.2. Template** 17](#_Toc447897892)

[**8.4.3. Command** 17](#_Toc447897893)

[**8.4.4. Observer** 17](#_Toc447897894)

[**9. Usage** 18](#_Toc447897895)

[**9.1. Installation Requirements** 18](#_Toc447897896)

[**9.2. Building requirements** 18](#_Toc447897897)

[**10. Testing** 19](#_Toc447897898)

[**10.1. Development Testing** 19](#_Toc447897899)

[**10.2. External Testing** 19](#_Toc447897900)

**2. List of Figures**

[**Figure 1:** UI sketches provided by client 8](file:///C:\Users\jermenkoo\Documents\GitHub\audava_meditation\Documents\team52_final_report.docx#_Toc447895760)

[**Figure 2:** UI sketch by our team 9](#_Toc447895761)

[**Figure 3:** description of main features of native vs hybrid development (IBM, 2012) 13](#_Toc447895762)

[**Figure 4:** Splash screen during loading 14](file:///C:\Users\jermenkoo\Documents\GitHub\audava_meditation\Documents\team52_final_report.docx#_Toc447895764)

[**Figure 5:** Initial screen – media player 14](file:///C:\Users\jermenkoo\Documents\GitHub\audava_meditation\Documents\team52_final_report.docx#_Toc447895763)

[**Figure 6:** Application menu 15](file:///C:\Users\jermenkoo\Documents\GitHub\audava_meditation\Documents\team52_final_report.docx#_Toc447895765)

[**Figure 7:** Settings screen 15](file:///C:\Users\jermenkoo\Documents\GitHub\audava_meditation\Documents\team52_final_report.docx#_Toc447895766)

[**Figure 8:** Categories (All) screen 16](file:///C:\Users\jermenkoo\Documents\GitHub\audava_meditation\Documents\team52_final_report.docx#_Toc447895767)

[**Figure 9:** Categories (Relationships) screen 16](file:///C:\Users\jermenkoo\Documents\GitHub\audava_meditation\Documents\team52_final_report.docx#_Toc447895768)

# **3. Abstract**

The application we were asked to develop is meant to provide people escape from every day’s routine by providing them with multiple guided and unguided meditation courses (or “interventions”, as defined by client). The target user of this application is a busy person, who is likely to be overwhelmed and stressed; thus this application will allow him to slow down and relax. The application should be simple to use, including the UI and the user should be able to see how much time he has left and be able to jump to any place he wants. Another possible functionality is to provide user reminders / notifications that he has broken his meditation streak and should start over.

# **4. Context**

## **4.1. Project Background**

The Mind Manifesto (MM) is a set of toolboxes to help people improve their well-being and counter the negative effects of anxiety and stress in today’s hectic environment. The toolboxes are meditation tracks focused on certain areas of our lives, such as work, health or relationships – areas which have the greatest influence upon us.

Our clients, Imran Ahmad and Terry Hall, have asked us to create the mobile version of The Mind Manifesto, so that the interventions are available offline and on-the-go. This is meant to accompany the main version of The Mind Manifesto, which is currently available as a web application.

## **4.2. Problem being solved**

According to (Lu, 1999), “stress has become one of the most serious health issues of the twentieth century – a problem not just for individuals in terms of physical and mental disability, but for employers and governments who incur significant financial damage.” (Hassard, et al., 2014) reports that in year 2012 the costs calculated in the EU were €20 billion a year, a figure which has almost doubled since then; (Blanding, 2015) estimates the figure for US up to $190 billion dollars.

It has been thoroughly researched that meditation is effective at mitigating the effects of stress - (Black, Manocha, Sarris, & Stough, 2011) report that “meditation, …, is a safe and effective strategy for dealing with work stress and depressive feelings”; (Astin, 1997) concludes “techniques of mindfulness meditation, …, may represent a powerful cognitive behavioural coping strategy”.

This is in conjunction what our application is aimed at – to help people in high-stress work environments, such as doctors or lawyers, cope with stress by listening to the series of guided meditation tracks.

# **5. Team Roles and Skills**

## **5.1. Jaromir Latal**

* **Team Role:** Team Leader / Project Manager, Lead Interviewer for Requirements, Lead Tester
* **Key Skills:** I am proficient in Python, as it is the first programming language I have taught myself and have used it during multiple projects and programming competitions. I have also familiarity with C++ and Java due to taking International Baccalaureate Computer Science course.
* **Prior Project Experience:**
  + Machine Learning project in Python (scikit-learn) to predict stock prices
  + Patients Database in Java & MySQL for a child doctor
  + Educational Math Game written in Java using Swing GUI library
* **Communication & Teamwork:** Debating for 3 years have given me the opportunity to respect others’ opinions and approach them neutrally, finding the best possible solutions for multiple parties. Working as a Software Engineer in the largest agricultural software company in Slovakia provided me a plenty of opportunities to improve my teamwork and communication skills.

## **5.2. Lambros Zannettos**

* **Team Role:** Lead Software Engineer & UI Designer
* **Key Skills:** I am proficient in Android development/Java (Android Studio), as I have completed several online courses leading up to the App Project (at Coursera and Udemy). I have also become familiar with C, HTML and JavaScript as part of the course.
* **Prior Project Experience:**
  + Browser-based educational game to teach Python written in JavaScript and the EaselJS set of libraries.
  + Machine Learning project to predict susceptibility to UV radiation, using MS Azure ML Studio.
* **Communication & Teamwork:** Prior to this course, I was a full-time self-employed musician/composer for several years. The nature of my work was highly collaborative, as creative projects often are. This along with being part in many amateur and professional bands over the years, and my national service experience before that, has taught me to listen to others with as little personal bias as possible, and always strive to make the best decisions to benefit the project/end result.

# **6. Requirements**

After being introduced to the project & client by Dr. Yun Fu, we discussed with our team possible scopes of the application. For this we researched current applications on the market (for Android and iOS) in order to prepare ourselves better for the upcoming meeting with the client. Our client sent us specifically three to have a closer look at – *Calm, Headspace and Insight Timer*.

## **6.1. Initial questions**

During the first meeting with the client we asked following questions to know the size and the scope of the project, as well as any preferences client has for the platform / technologies and design.

* What is exactly the project about?
* What platform would you like to target? Are there any preferences?
* Are there any technologies / languages / frameworks we are expected / asked to use?
* Details about the application and its functionality:
  + Do you have any specific UI design in mind?
  + What features are required for the core functionality?
  + What are the additional features we might implement?
  + Should it be on-line / off-line application? (e.g. tracks stored remotely)
  + Should the application feature in-app purchases as well?
  + Should the user have some settings available?
  + How many songs do you plan the app to hold? (relevant for off-line version)

## **6.2. List of requirements**

Based on the client’s answers to the questions, we drafted a list of requirements in MoSCoW (Must-Should-Could-Would) style, splitting the requirements into 4 categories based on their importance for the success of the project. After meeting with HCI Teaching Assistant Aisling O’Kane we finalised it and have it approved by our client.

**Must have -** **What must be delivered, i.e. it is essential for this phase?**

* Capability to play mp3 files stored locally on the device, including common media functions like play/stop/pause.
  + Complete: Media player and all its functions work as intended.
* Ability to detect connected headphones as the content is to be listened to via headphones only.
  + Complete: Media player displays a message to the user that for best experience headphones should be plugged in.
* Minimal, simple and intuitive UI design.
  + Complete: The design was rated well by both client and users.

**Should have - What should be delivered as a high priority but not essential?**

* A reminder function by which the app reminds the user to take some time to meditate. This could be either at random times or at specific times each day.
* Skip forward/backward buttons that skip a set number of seconds.
  + Complete: The skip time in seconds can be set in settings.
* Remembering last place left off in a session (if it was closed before completion).
* A big countdown timer signifying the time left in a session.
  + Complete: The users is being shown the time left on the music player screen.

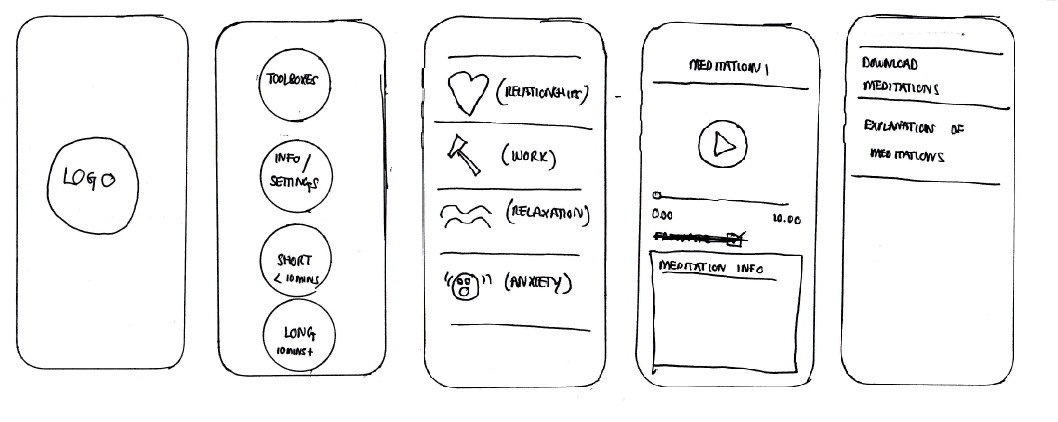
**Could have - What could be delivered if there was available time / budget / resource?**

* A store part of the app, where the users can buy and download new content.
  + Incomplete: Pushed back into next version after discussion with client.
* Translatable (this is easy to implement so could also belong in the Must list above).
  + Complete: Application is localised into user’s system language.

**Would have - What would be delivered if time / budget / resource was unlimited?**

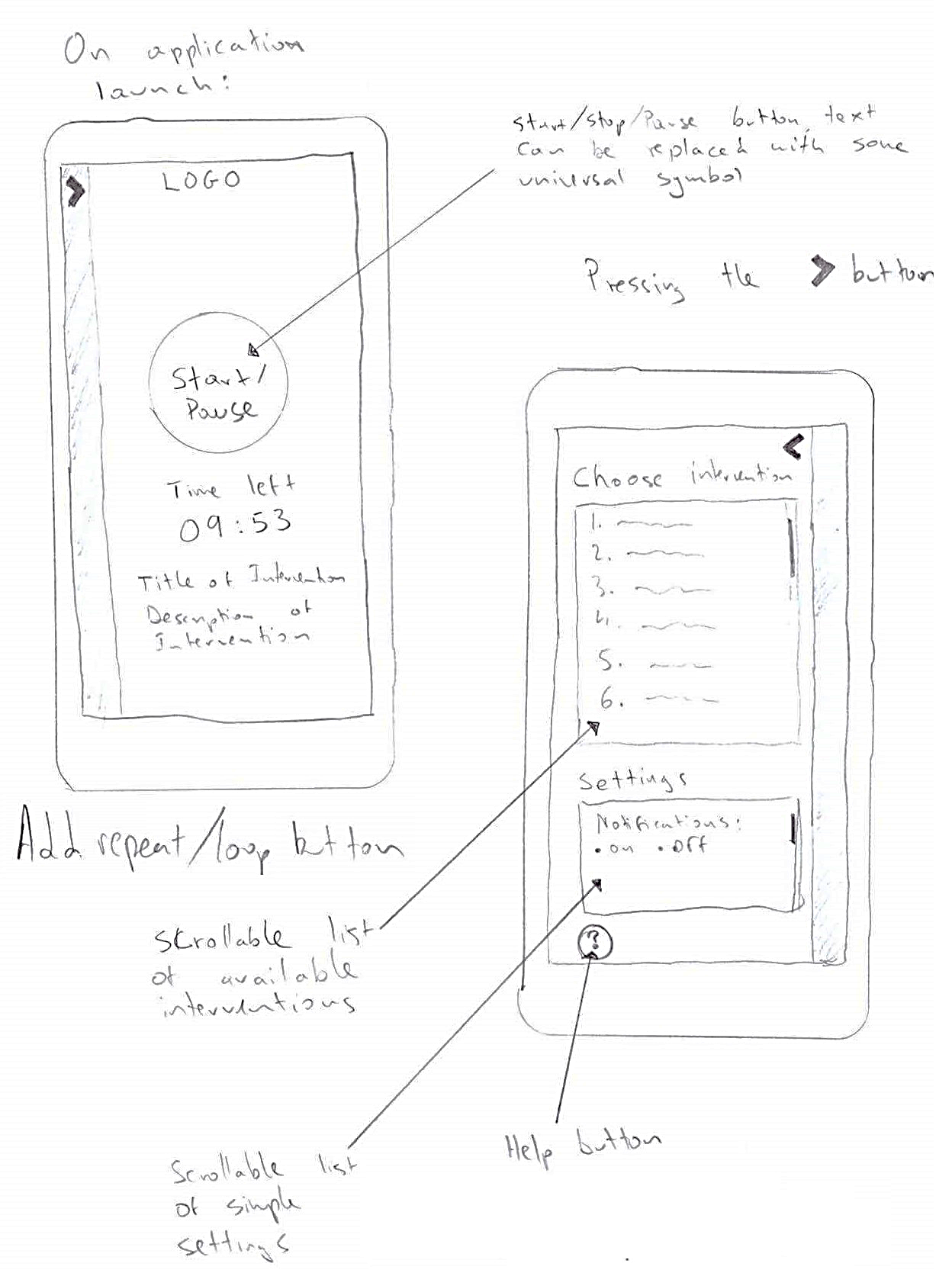
* Intelligent reminder that detects habits and breaks them.
  + Incomplete: Not possible in the time-frame given.
* Payment integration for new interventions in the application
  + Incomplete: Not enough time and issues present with testing in Google Play Store.

## **6.3. Design**

The client has asked us to deliver a minimal, simple and clean UI, so that the application will not be cluttered and every first-time user will immediately know where to click. We created a functional UI prototype independently and compared it to the client’s sketches, settling on the final design.

**Figure 1:** UI sketches provided by client

(from left: splash screen, main menu, toolboxes, media player, settings)



**Figure 2:** UI sketch by our team

Both UI mock-ups are quite similar **–** the menu is hidden from the user not to obstruct his view and design of media player shares the idea of a big button along with time remaining being shown in the session. We agreed on adding settings screen as a separate option in the menu rather than having shown all the settings there. Based on this we settled on the final design, which is shown later in this report, already implemented.

## **6.4. Use Cases**

We carried out use case analysis in our project in order to better understand the system we will be building. It also allowed us to further look at our requirements and whether they can be refined better. We did so by illustrating interaction between the system (application) and its only actor – user.

|  |  |
| --- | --- |
| Use Case (UC1) – Application is opened for the first time | |
| Primary Actor | User |
| Precondition | The application has not been opened yet. |
| Postcondition | The main menu appears and the tutorial screen will not appear anymore. |
| Main paths | 1. User opens the application. 2. Splash screen appears for a few seconds. 3. Brief tutorial appears describing the application to the user and how to navigate in the application. |
| Alternative paths | None |

|  |  |
| --- | --- |
| Use Case (UC2) – Application is opened | |
| Primary Actor | User |
| Precondition | The application has not been running. |
| Postcondition | The main menu appears. |
| Main paths | 1. User opens the application. 2. Splash screen appears for a few seconds. 3. The main menu appears. |
| Alternative paths | None |

|  |  |
| --- | --- |
| Use Case (UC3) – A new intervention is chosen | |
| Primary Actor | User |
| Precondition | The application is running. |
| Postcondition | The chosen intervention starts playing. |
| Main paths | 1. User selects the intervention’s category. 2. User selects a specific intervention. 3. The menu is hidden and the player screen appears. 4. The chosen intervention starts playing. |
| Alternative path #1 | 4. a) The intervention track cannot be loaded.  4. b) User is notified and returned back into the selection screen. |

|  |  |
| --- | --- |
| Use Case (UC4) – Intervention is paused or resumed | |
| Primary Actor | User |
| Precondition | An intervention has been loaded successfully and is playing. |
| Postcondition | An intervention is either paused or resumed. |
| Main paths | 1. User selects the pause button in the player screen. 2. The intervention stops playing. |
| Alternative paths | 2. a) The intervention has been paused previously.  2. b) The intervention is resumed. |

|  |  |
| --- | --- |
| Use Case (UC5) – Skipping to a certain part of intervention by scrubbing | |
| Primary Actor | User. |
| Precondition | The media player screen is active and an intervention is loaded. |
| Postcondition | The time position of the track is changed. |
| Main paths | 1. User scrubs on the position slider using a finger. 2. Once the finger is released, the track jumps to the position. 3. Playback continues from the new position |
| Alternative paths | 3. a) The intervention was paused before scrubbing. The playback does not continue until resumed. |

# **7. Work Distribution**

## **7.1. Jaromir Latal**

* Add error handing
  + Complete: Error handling has been added to prevent user from seeing errors / notifications; all errors are being logged.
* Add usage logging
* Fix bugs and issues found during the development
  + Complete: All reported issues on GitHub have been closed after being successfully solved.
* Refactor code
  + Complete: During the development we strictly adhered to OOP principles and made use of multiple design patterns.
* Test the application
  + Complete: The application has been subjected to multiple types of testing, including the issues and bugs found during development.

## **7.2. Lambros Zannettos**

* Develop music player functionality
  + Complete: Implemented media player from scratch with cueing by desired time in seconds.
* Intelligent intervention filtering based on the category
  + Complete: After a category is chosen user sees only interventions belonging to that certain category.
* Getting metadata from interventions
  + Complete: Information about intervention are being read from an ID3 tag in the media file instead of being somewhere hardcoded.
* Changeable settings
  + Complete: User’s settings are saved persistently on the phone.

## **7.3. Shared**

* Produce bi-weekly reports
  + Complete: Bi-weekly reports have been filled in and submitted on time.
* Create an initial UI mockup
  + Complete: UI mockup has been completed based on the individual mockups created by each of us.

# **8. Technical Design**

## **8.1. Technology chosen**

After having agreed with the client that we want to target **Android** platform due to number of devices on the market, we had to decide which technology to use. We considered the following two possibilities: **Android Studio** (native Android SDK – Software Development Kit) and **3rd party SDKs** (such as **Cordova** or **Xamarin**).

**Android Studio** is the IDE which provides access to Android SDK - a modified version of Java including Android features and Google APIs - used to develop native Android applications.

**Xamarin** is a cross-platform mobile development tool used to develop applications both for Android and iOS (in C#). Xamarin apps consist of two layers:

1. **iOS / Android** – containing the UI and application layers (written in C#)
2. **Core Library** – containing shared code such as business logic

**Cordova** is a mobile cross-platform application development framework for **Android**, **iOS**, **Windows** **Phone** and more. Development is done in HTML5, CSS3 and Javascript instead of platform specific APIs (such as the Android SDK), however it allows using native device APIs providing access to device’s camera, file system and more.



**Figure 3:** description of main features of native vs hybrid development (IBM, 2012)

We decided to utilise the native **Android SDK**. The reasons were the following:

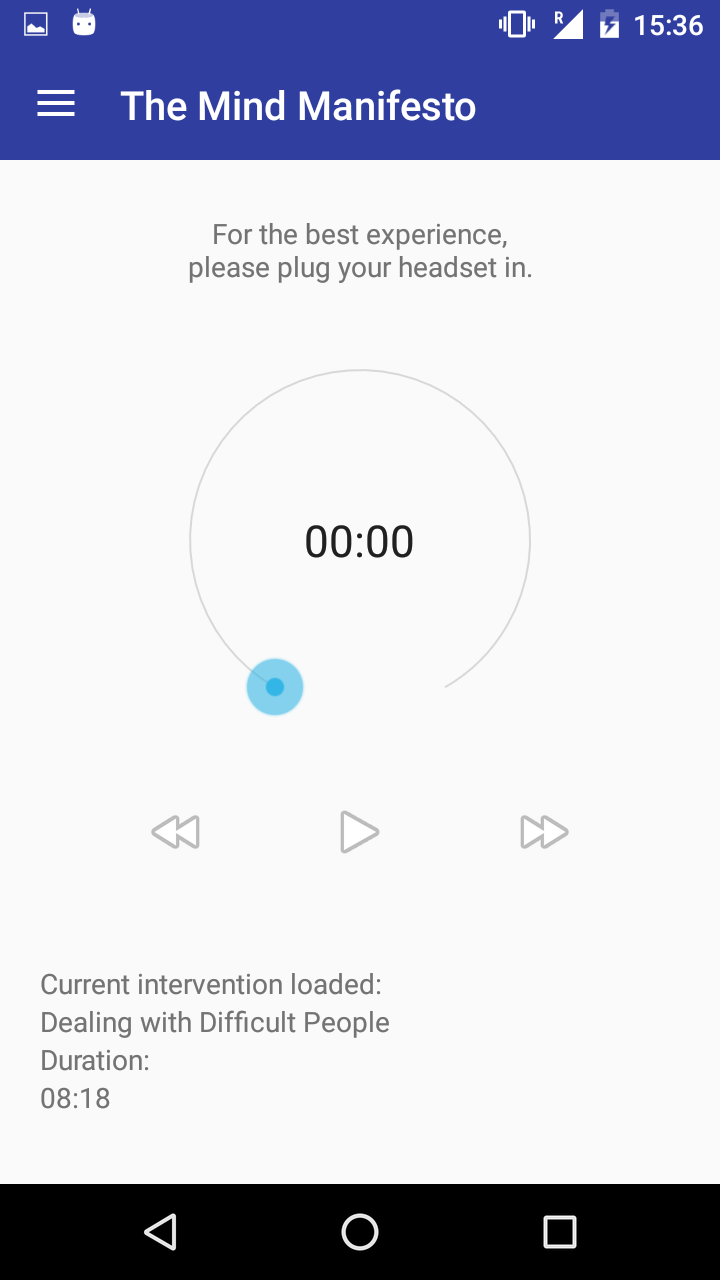
1. Using native APIs guarantees the look-and-feel will be the same on every device for the given platform.
2. Since the application is meant to be used for a longer amount of time (for meditation), the web application could run out of the memory, possibly resulting in bad user feedback. (http://venturebeat.com/2013/04/17/linkedin-mobile-web-breakup/)
3. Going native guarantees future compatibility, as Android Studio always adapts to the newest **Android SDK**, however when a new version of Android is released, cross-platform development tools do not support it until an update is released.

## **8.2. Structure of the Application**

Flowchart

Code hierarchy

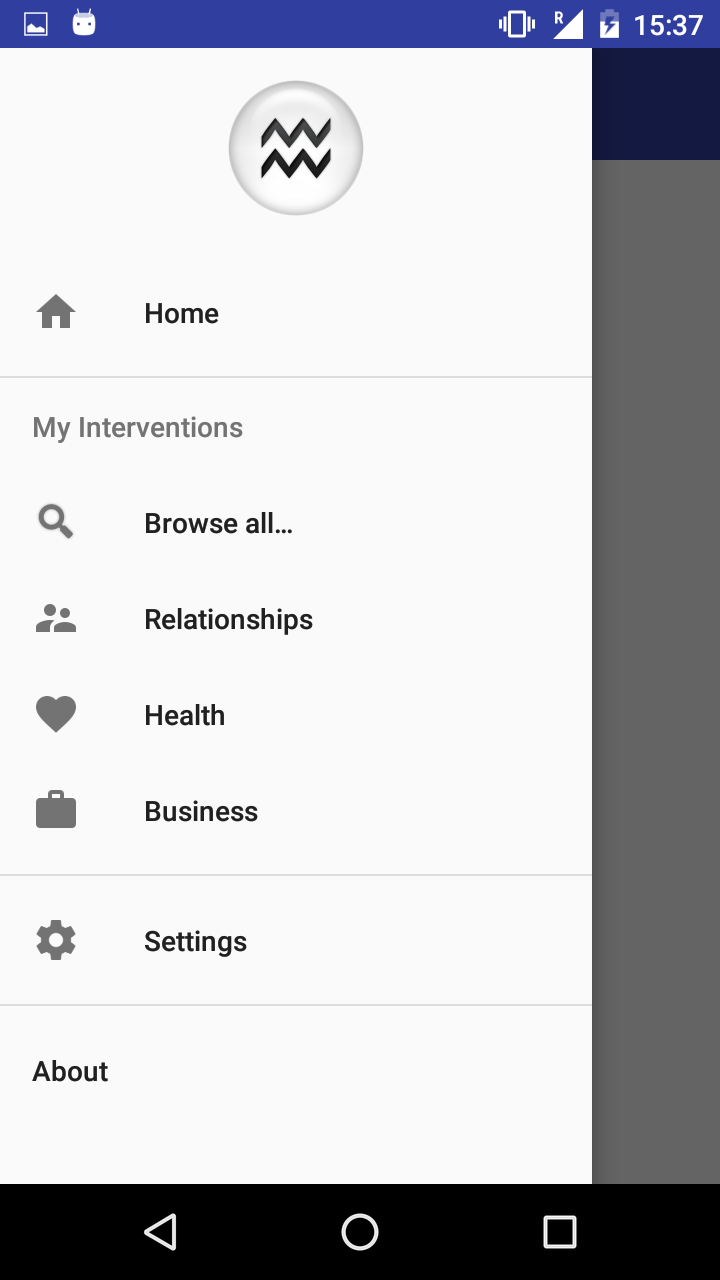
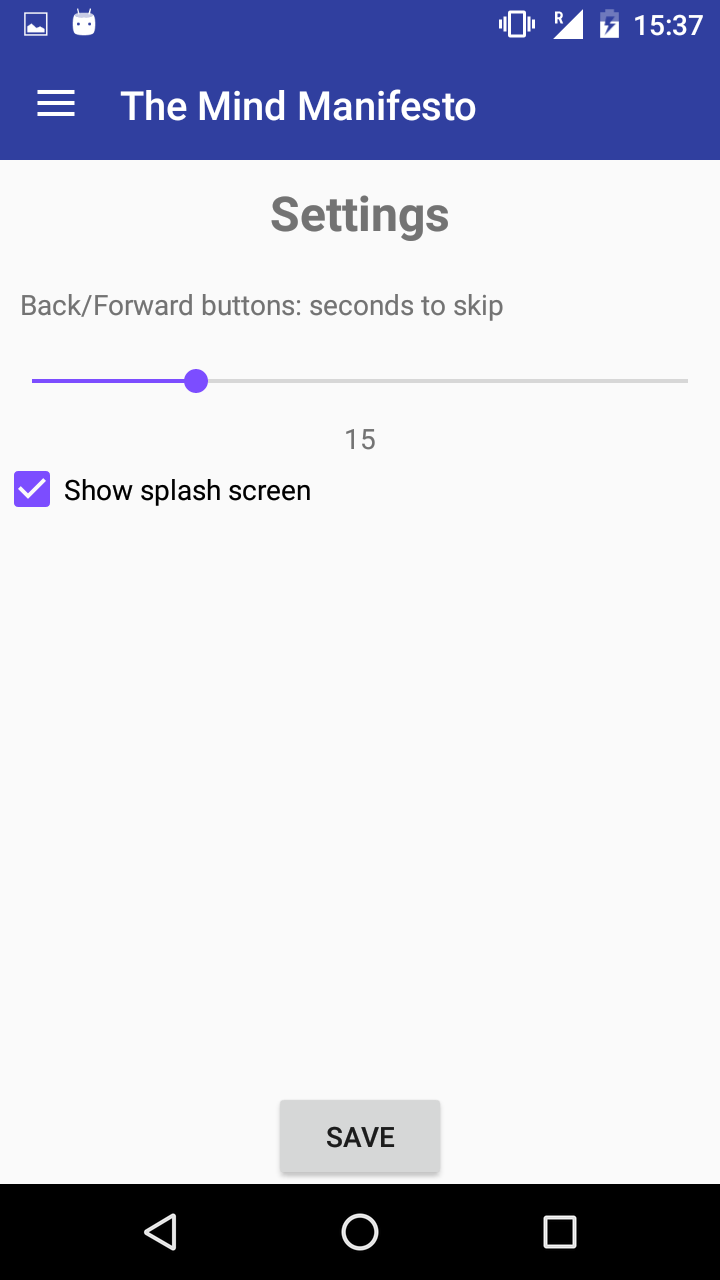
## **8.3. User Interface**

We designed our user interface so that it is intuitive to use – it consists out of simple design blocks (buttons, sliders, text views) stitched together to provide the best experience. We wanted the app to be easy to use for everybody without having to read a manual or wander around all the options for 15 minutes.

**Figure 5: :** Initial screen – media player

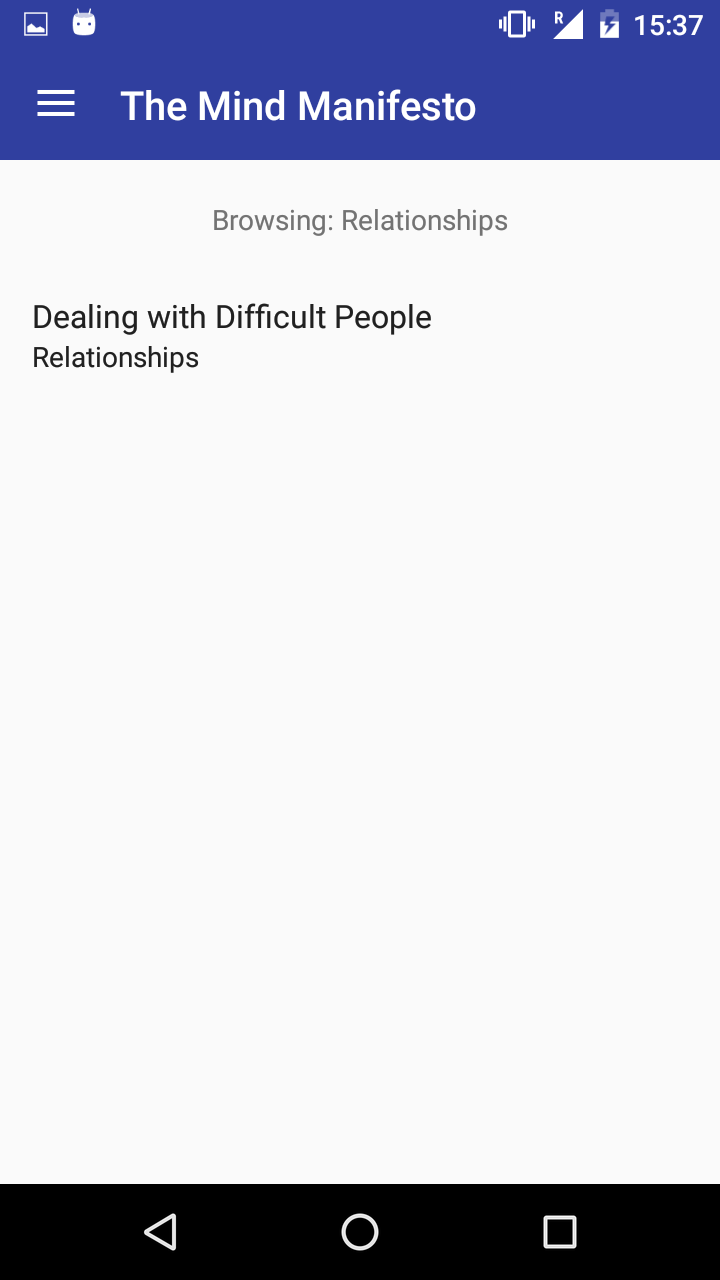
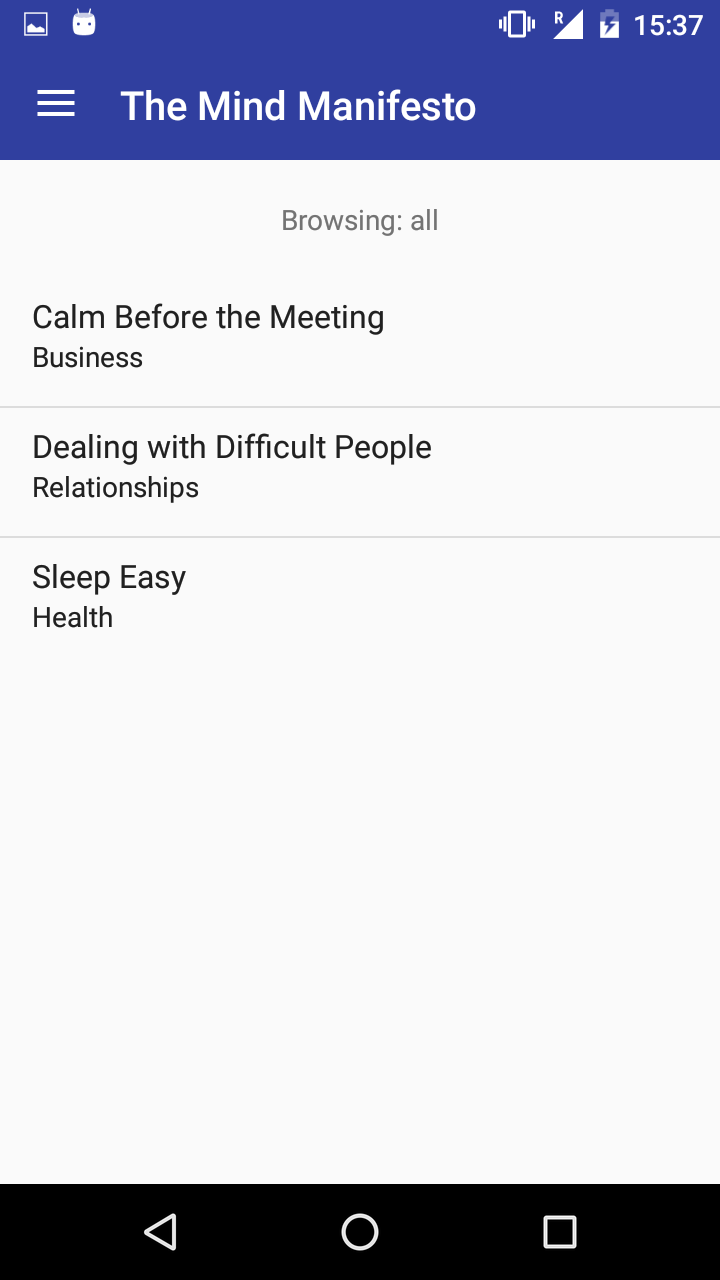
When the application is launched, the splash screen is shown (Figure 2) during which the files needed for application launch are being loaded. Afterwards the user is shown directly the media player (Figure 3) with one of the meditation tracks loaded, so that he can try out the application directly.

**Figure 4:** Splash screen during loading

The menu can be accessed by swiping from the right edge of the screen. It contains basic options, namely main screen (where the media player is located), interventions available in the current version (split into their own categories), settings screen in which the user can alter the amount of time he wants to cue forward or backward and whether the splash screen should be shown.

**Figure 6:** Application menu

**Figure 7:** Settings screen

In order to play an intervention, user has to choose the category he wants to listen to and then choose a specific intervention out of the list. There are two types of categories: all, where all the interventions in the application are available or specific, such as Relationships, Health or Business, where only interventions belonging to this specific category are shown.

**Figure 8:** Categories (All) screen

**Figure 9:** Categories (Relationships) screen

## **8.4. Design Patterns**

### **8.4.1. Singleton**

The Singleton pattern restricts the instantiation of the class to only one object, meaning only one instance of a class can be created, often being accessed globally.

Singleton has been implemented in our application in the MediaPlayerSingleton class. We decided to use so-called eager initialisation of Singleton, because our application will always need an instance (so that the MediaPlayer can play interventions in the applications). Moreover, the cost of creating one instance is not too expensive, which further affirmed our decision.

Using this design pattern ensured that the MediaPlayer instance always exists and that after changing screens the reference to the instance still exists, which is better rather than creating a completely new instance of MediaPlayer when a different meditation track is chosen.

### **8.4.2. Template**

Template pattern is a behavioural pattern design, which defines the skeleton of the algorithm and defers some steps to the subclasses – they can re-define certain steps as needed without changing the algorithm’s structure (which is common for all of them).

We made use of this pattern when creating items belonging to each category. While each category should have different items (some of them can be the same, e.g. a meditation track will be both in its own category and in all categories), the behaviour should be the same – when clicked the music player screen should be shown and the song played.

Usage of this design pattern allowed us to remove duplicate code which would have been otherwise needed for every item in the menu.

### **8.4.3. Command**

Command pattern and where we use it

### **8.4.4. Observer**

The Observer pattern is a behavioural pattern - when an object is modified, all other object depending on it (called observers) are notified automatically by calling one of their methods.

Observer pattern in our application is implemented in the HeadphoneStateReceiver class, which extends the built-in BroadcastReceiver. Once the headphones are plugged in, the HeadphoneStateReceiver receives notification from ACTION\_HEADSET\_PLUG listener and shows a notification to the user – to notify him that the headphones have been plugged.

# **9. Usage**

## **9.1. Installation Requirements**

The application can be installed on any Android device whose Android version is at least 4.0.3 (Ice Cream Sandwich) or higher. We decided to support all devices from this version onwards to have the highest market share possible (4.0.3 and newer devices account for 97.3% all Android devices (Google, 2016)), while making use of the newest features and libraries in the Android SDK.

Since the application uses features available only for Android 5.0 and newer, we make use of official Android Support Library in order to maintain compatibility with devices with Android version 4.0.3-4.4. (Google, 2016)

## **9.2. Building requirements**

In order to build the application, simply open either *audava\_meditation.iml* or *TheMindset.iml* using Android Studio. Once the project is loaded, the application should be built automatically using Gradle build system, which will compile all the dependencies of the application and run tests. Provided the testing has finished without any error, an *.apk* (Android package) file is built, which can be then installed and used on any supported Android device (outlined above).

# **10. Testing**

## **10.1. Development Testing**

During development we tested our application on multiple devices to ensure that it works as needed. We used following devices during testing:

* Sony Xperia Z3 Compact, model number D5803, OS version: Android 5.1.1
* Huawei Ascend, model number: Y330, OS version: Android 4.2
* Motorola Moto G2, model number: XT 1068, OS version: Android 6.0

We chose variety of devices with different OS versions to test the application on the widest range of devices as possible.

## **10.2. External Testing**

During and after development Jaromir conducted tests throughout our development, asking third party people with no previous knowledge about the application to try it out and provide their opinion – score out of ten in following categories: Overall, User Interface, User Experience, Ease of Use and Problem Encountered.