BPM Detector (Android)

Requirements Document

Duc Dao
CPE 491, Spring 2017
California Polytechnic State University, San Luis Obispo

Abstract

The BPM Detector is an Android application that detects the tempo or beats-per-minute (BPM) for a band conductor's conducting. Generally, the conductor moves their arms in a predictable pattern in order to maintain the band's synchronization, acting similarly to a metronome. The purpose of this application is to provide a tool for concert and marching band conductors to detect their precise tempo while conducting. This will help them determine how well they are coordinating the band and whether they need to adjust their conducting.

Contents

1 Introduction

1.1 Purpose

This document is created to discuss the specifications of the BPM Detector, an Android application that detects the beats-per-minute of a conductor's tempo while conducting. The main purpose of the BPM Detector is to aid conductors in seeing how well they are conducting the band and determine whether they need to make adjustments or not. This will benefit marching bands because these types of bands are generally more sensitive to tempo inconsistencies compared to other types of bands.

1.2 Document Conventions

This document is broken up into six main sections with the following headings:

1. Introduction
2. Overall Description
3. External Interface Requirements
4. Functional Requirements
5. Nonfunctional Requirements

These main sections each include subsections and minor subsections, which may contain descriptions, bulleted lists, and tables.

1. Main Section

1.1 Subsection

1.1.1 Minor Subsection

Body text

1.3 Intended Audience

The intended audience for this document includes:

- Developer
- Advisors
- Application Testers

1.3.1 Developer

This document will serve as a guide for myself, the developer of BPM Detector. It will primarily be used to aid in my design and implementation of this project.

1.3.2 Advisors

Advisors include:

- My senior project advisor, Professor Franz Kurfess
- All other resources that aid in my project which may include other professors, students, and other developers

1.3.3 Application Testers

Application Testers are users that have knowledge and capability of conducting a song. This includes, but is not limited to:

- Band directors
- Drum majors, student conductors
- Professors in the Music department

2 Overall Description

2.1 Background

In the context of music ensembles, conducting is the act of directing the group in order to synchronize various components of the band and the performance. Members rely on the conductor to initiate the piece/song that they are performing, set the beats-per-minute or tempo of the song, cue points of interest in the piece, and other musical duties.

With regards to the tempo, there are no applications to automatically detect the conductor's BPM while they are conducting. Existing solutions on the Google Play Store have users tap on the screen to manually calculate the BPM and metronome applications only sound off the BPM by setting a certain tempo, or listening to the environment.

2.2 Product Functionality

BPM Detector's will perform one major function: to automatically detect the conductor's tempo while they are conducting.

2.3 User Class and Their Characteristics

BPM Detector will be used by one general user class: conductors. Conductors' experiences varies from person to person but all of them generally have knowledge of:

- Conducting patterns
- Feeling of tempos
- Technicalities of the song they are conducting

2.4 Operating Environment

BPM Detector will be an Android application running on a smartwatch with at least Android Wear 1.5. The smartwatch requires a Bluetooth connection to an Android phone running at least 6.0 Marshmallow. The smartwatch must have the follow specifications:

- Central Processing Unit (CPU): 1.2 GHz quad-core Qualcomm Snapdragon 400 CPU (APQ 8026) or greater
- Graphics Processing Unit (GPU): Adreno 305 with 450MHz GPU or greater
- Memory: 2 GB or greater
- Storage: At least 100 MB of free storage
- Connectivity: Bluetooth 4.0 or greater

3 External Interface Requirements

3.1 User Interfaces

Users of BPM Detector will interact with two user interfaces (UI); the UI of the Android Wear smartwatch, and the UI of the Android smartphone. On both devices, users will be able to:

- Start the BPM detection
- Stop the BPM detection
- View past detections

BPM Detector will follow Material Design, a design language developed by Google.

3.2 Hardware Interfaces

BPM Detector will be installed on the smartwatch and on the Android smartphone it is connected to. Bluetooth connectivity on the phone must persist during the life of a detection session in order for the smartwatch to send information to the smartphone.

3.3 Software Interfaces

Both devices will have the capacity to initiate the BPM detection. During detection, the smartwatch must be able to interpret data from the sensors of the smartwatch and output a number representing the BPM. The smartwatch must interface with the smartphone in order for the smartphone to receive the BPM. This will allow the conductor to see the tempo they are conducting at.

4 Functional Requirements

BPM Detector functional requirements include the following use cases:

1. Starting BPM detection
2. Stopping BPM detection
3. Viewing past detections

4.1 Starting BPM Detection

Use Case ID	1
Name	Starting BPM Detection
Description	The act of starting the tempo detection.
Preconditions	User must have the smartwatch on their wrist and the application on the phone launched.
Postconditions	None
Priority	High
Frequency of Use	High
Normal Course	User commences tempo detection.
Actor Action 1. User selects ”Start” on either the smartwatch or smartphone.	**System Responses** 2. Smartwatch receives initiation and tells sensors to retrieve information in real-time. 3. Smartwatch interprets sensor information and calculates a BPM number. 4. Smartwatch sends BPM number to smartphone.

4.2 Stopping BPM Detection

Use Case ID	2
Name	Stopping BPM Detection
Description	The act of stopping the tempo detection.
Preconditions	User started the application and initiated detection. Smartwatch is currently detecting BPM.
Postconditions	Smartwatch has stopped detecting BPM.
Priority	High
Frequency of Use	High
Normal Course	User ceases tempo detection.
Actor Action 1. User selects ”Stop” on either the smartwatch or smartphone.	**System Responses** 2. Smartwatch receives cease selection and tells sensors to stop retrieving information in real-time. 3. Smartphone saves information of the duration of the detection as history for future viewing.

4.3 Viewing Past Detections

Use Case ID	3
Name	Viewing Past Detections
Description	Viewing detections user made in the past.
Preconditions	User has done at least one detection and it saved correctly. Viewing past detections can only be done on the smartphone.
Postconditions	None
Priority	Low
Frequency of Use	Medium
Normal Course	User views past detections.
Actor Action 1. User selects ”View History” on smartphone. 3. User selects which past detection they want to view.	**System Responses** 2. Smartphone receives selection and takes user to a listing on past detections. 4. Smartphone displays past detection the user has selected and displays information associated with that detection.

5 Non-Functional Requirements

5.1 Performance Requirements

All actions performed in the application must take a reasonable amount of time; the faster, the better. The goal is to keep all requests made (starting, stopping, viewing) under 5 seconds in terms of response time. To achieve this, the developer must implement the application with consideration to the smartwatch’s limited resources.

5.2 Safety Requirements

During detection, the smartwatch’s and smartphone’s temperature cannot exceed 110 F. To prevent this, the developer must be able to utilize the smartwatch’s limited resources efficiently.