Revision History

Date	Developer(s)	Change
October 6, 2017	Aidan/Sullivan/Karlo	Initial creation of document.
December 5, 2017	Sullivan	Finalized

SE 3XA3: Module Guide
Shuffle

Team 16, AsK Studios
Aidan Schonewille, schonea
Sullivan Stobo, stobos1
Karlo Delalic, delalik

December 6, 2017

Contents

List of Tables

List of Figures

1 Introduction

The purpose of this document is to outline the various modules that the project 'Shuffle' is divided into. The practice of dividing software is to various modules is very common as it allows the program to be viewed in smaller parts, which can make it easier to understand. It also makes it easier to trace issues and add features.

2 Anticipated and Unlikely Changes

This section lists any changes that may be made to the system. They are categorized according to likeliness. Anticipated changes list likely changes, while Unlikely changes list unlikely changes

2.1 Anticipated Changes

These changes would likely require only changing one module, making them simple to implement without massively altering the system.

AC1: Allowing the webpage to work on mobile devices.

AC2: Changing the colouring and layout of the webpage.

AC3: Altering where components are on the webpage.

AC4: Using a different search term as the default search term.

2.2 Unlikely Changes

These changes would be more difficult to implement as it would involve altering many of all of the modules. It is intended that these changes will not be made to the system.

UC1: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

UC2: There will always be a source of input data external to the software.

UC3: We will always be using YouTube to provide music videos. No other service will be used.

UC4: The project will always use React.js and the YouTube APIs.

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: Behaviour-Hiding Module

M3: Software Decision Module

M4: App Module

M5: Controls Module

M6: Search Module

M7: Search Bar Module

M8: Top Bar Module

M9: Video Module

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	App Module Search Bar Module Top Bar Module Video Module
Software Decision Module	Search Module Controls Module

Table 1: Module Hierarchy

4 Connection Between Requirements and Design

The design of the system Shuffle is to provide a easy way to shuffle play music video from YouTube. This is reflected by the requirements specified in the SRS document. The module outlined below are divided in what we believe to be the best and simplest way to satisfy the requirements.

5 Module Decomposition

Modules are decomposed according to the principle of "information hiding". The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. Also indicate if the module will be implemented specifically for the software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented. Whether or not this module is implemented depends on the programming language selected.

5.1 Hardware Hiding Modules (M1)

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

5.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

5.2.1 App Module (M4)

Secrets: -

Services: Provides the structure and layout of the webpage.

Implemented By: Shuffle

5.2.2 Controls Module (M5)

Secrets: Playing/Pausing Videos and going to next and previous videos.

Services: Allows the user to control video playback.

Implemented By: Shuffle

5.2.3 Search Bar Module (M7)

Secrets: -

Services: Provides a place for the user to input search text.

Implemented By: Shuffle

5.2.4 Top Bar Module (M8)

Secrets: -

Services: Provides a component to contain the search bar, logo and the displays what is currently playing.

Implemented By: Shuffle

5.2.5 Video Module (M9)

Secrets: Video playback

Services: Contains the video player and controls what video is playing.

Implemented By: Shuffle

5.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

5.3.1 Search Module (M6)

Secrets: YouTube Data API call

Services: Finds a list of video based on a search query.

Implemented By: Shuffle

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	**Modules**
R1	M4, M9
R2	M3, M4, M9
R3	M4, M6, M9
R4	M5, M9
R5	M4, M8, M9
R6	M4
R7	M4
R8	M3, M4
R9	M3
R10	M9

Table 2: Trace Between Requirements and Modules

AC	**Modules**
AC1	M1
AC2	M4
AC3	M4, M8, M5, M9
AC4	M3, M6

Table 3: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. **?** said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

Figure 1: Use hierarchy among modules