Lesson 16 – In-Class Worksheet

Jacob Dunn – M3

Documentation Statement: None

Your newly-launched start-up company wants to propose developing a fuzzy robot that is smart, interactive, educational, and helpful. This robotic friend will be marketed towards children ages 4 to 10 to interact with user (i.e. answer questions and express feelings), play music, track daily routines, and play games.



www.woobo.io

As the senior software engineer, you are responsible for creating the proposal for the software development of one of the above desired capabilities (interact with user, play music, track daily routines, and play games) to include outlining the requirements, functionally decomposing the desired final product, and providing a structure chart.

Part I – Extracting the Requirements

Provide a set of at-least ten enumerated requirements for your system. Be sure the identified requirements are correct, feasible, necessary, unambiguous, and verifiable.

Part II – Functional Decomposition

Break the desired final software product into a small set of high-level steps (tasks) that need to be accomplished. Revisit each high-level task and break it into smaller steps until the tasks are small enough to implement directly.

Part III – Structure Chart

From the above functional decomposition, create a structure chart where links denote a task referencing, or using another.

Desired Capability: Play Music

Part I:

1. Initialize screen with no song playing
2. Output “Previous”, “Play/Pause”, “Next”, “volume down” and “volume up” to touchscreen
3. When play is pressed start playing downloaded song
4. Add a “shuffle” and “repeat” button to the upper right corner of the screen
5. Add an “Up Next” button to the bottom right corner
6. Print name of song and artist to the screen above the play button for each new song
7. Constantly check if Play/Pause, next or previous is hit
8. Function for holding songs alphabetically that plays the next song upon hitting next button or the last song if previous is hit. This function shuffles library when shuffle is hit and replays the same song if repeat is pressed, and takes over the screen to show the next 10 songs that will be played when Up Next is pressed
9. Function to stop the music and remember the time it’s stopped at when pause/play button is pressed
10. Reset to initialize screen once last song has finished playing.

Part II:

* main
* Call Initialize
* Call Input\_Status (Repeat until finished)
* Call Song\_List
* Call Last\_Song\_Over
* Initialize
* Set song\_playing to nothing
* Set Shuffle, Repeat and Up Next to off
* Sets all songs in Song\_List to alphabetical
* Draw the buttons to the screen
* Input\_Status
* Check if pause/play is pressed
* Check if previous is pressed
* Check if next is pressed
* Check if shuffle is pressed
* Check if repeat is pressed
* Check if up next is pressed
* Pause\_Play
* If music was playing, then pauses

1. Stops music
2. Remembers song and time

* If music was paused, starts playing

1. Plays music at same point it left off

* Shuffle
* Randomly orders songs in Song\_List
* Repeat
* Doesn’t go to next song when song ends
* Next
* Skips to the next song in Song\_List
* Previous
* Plays the previous song in Song\_List
* upNext
* Display next 10 songs to be played in list to the screen
* Song\_List
* Begins with first song in the alphabet
* Orders songs in list from 0 (Blank) to NUM\_SONGS
* Last\_Song\_Over
* Set song\_playing to nothing
* Set Shuffle, Repeat and Up Next to off
* Sets all songs in Song\_List to alphabetical
* Draw the buttons to the screen

Part III:

Diagram, table

Description automatically generated