Design Document

Version 2.0 - Lay out - April 2, 2019

EECS 2311 Submission: Group 12

Mohammed Nasiful Haque
Amaan Vania
Tony Ly

Revised	By
April 2, 2019 - layout	Tony Ly

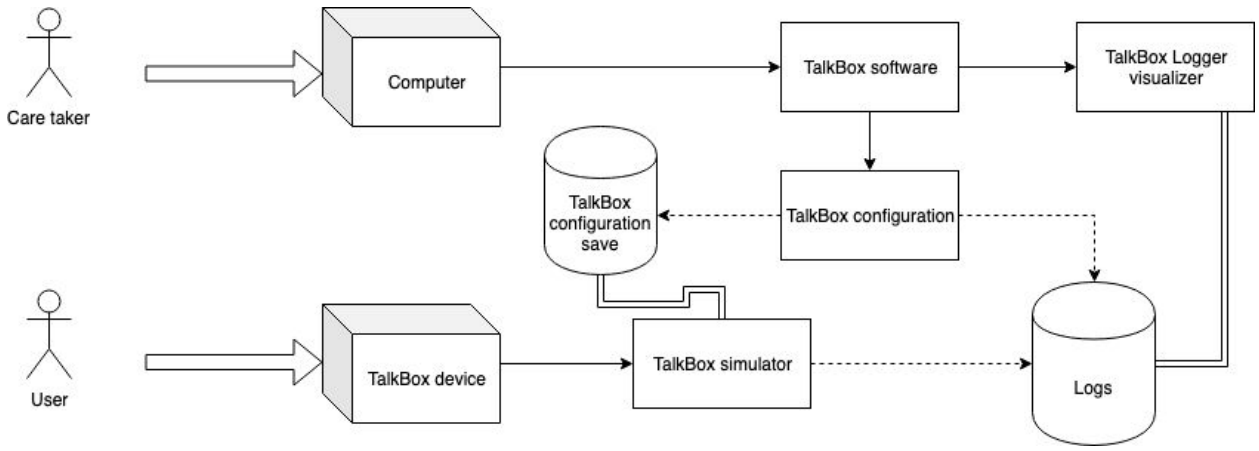
Purpose

The purpose of this document is to outline the design of the TalkBox software. This document will summarize how users interact with the software and how the software works. It is assumed that an experienced developer will be able to read this document and understand the context of the design and how the whole system comes together. In this document, there will be three main diagrams, system diagram, sequence diagram, and class diagram.

System diagram

The System diagram gives a basic overview of how the larger scale components such as the simulator and configuration act.

In the diagram, the cubes represent physical devices, and the caretaker accesses the TalkBox software via their computer. The square boxes represent the parts of the TalkBox software, such as the TalkBox configuration that the caretaker will use. The cylinders are files and the pipelines represent a pipelining file to software. Lastly, the dash lines represent the storing of information.

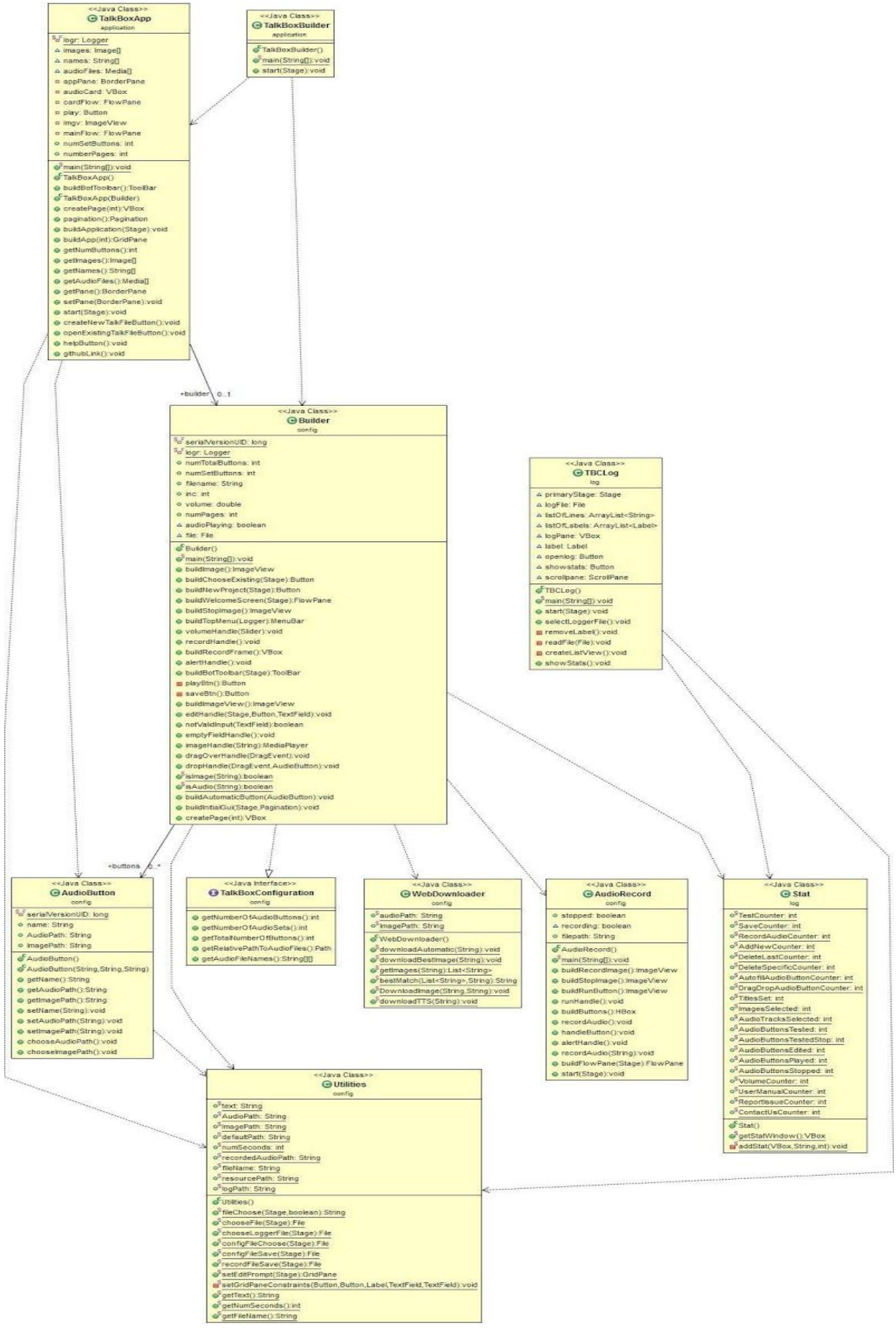


Class diagram

The class diagram highlights methods that are contained within each class and shows the deeper connections between the classes that make up the TalkBox applications software. It is to give a general overview of the codebase itself and also the relationships that exist between each of the objects.

The three main classes that cover the majority of the software requirements are the TalkBoxApp, the Builder and the TalkBoxLogger classes. The remaining classes are there to take some weight off the shoulders of these three main classes and obey the design principles and allow maintainability and scalability of the software. The classes were also divided to ensure that unit testing could be carried one each of those helper classes to prove the correctness of the main classes mentioned above.

A higher quality picture could be found at <https://i.imgur.com/vw6TS7L.jpg>.

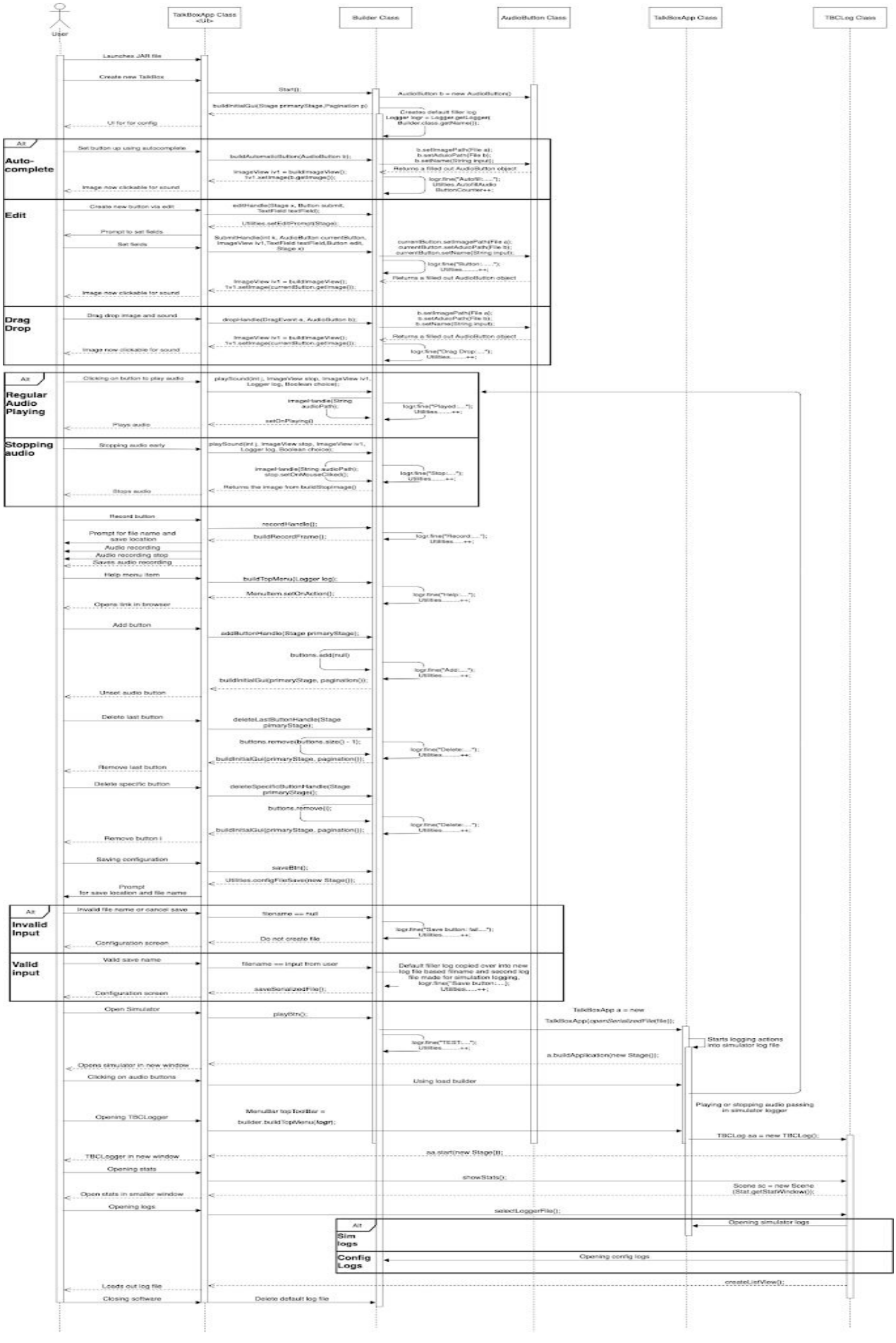

<<Java Class>>
TalkBoxApp
application
logr: Logger
images: Image[]
names: String[]
audioFiles: Media[]
appPane: BorderPane
audioCard: VBox
cardFlow: FlowPane
play: Button
imgv: ImageView
mainFlow: FlowPane
numSetButtons: int
numberPages: int
main(String[]):void
TalkBoxApp()
buildBotToolbar():ToolBar
TalkBoxApp(Builder)
createPage(int):VBox
pagination():Pagination
buildApplication(Stage):void
buildApp(int):GridPane
getNumButtons():int
getImages():Image[]
getNames():String[]
getAudioFiles():Media[]
getPane():BorderPane
setPane(BorderPane):void
start(Stage):void
createNewTalkFileButton():void
openExistingTalkFileButton():void
helpButton():void
githubLink():void
<<Java Class>>
TalkBoxBuilder
application
TalkBoxBuilder()
main(String[]):void
start(Stage):void
+builder
0..1
<<Java Class>>
Builder
config
serialVersionUID: long
logr: Logger
numTotalButtons: int
numSetButtons: int
filename: String
inc: int
volume: double
numPages: int
audioPlaying: boolean
file: File
Builder()
main(String[]):void
buildImage():ImageView
buildChooseExisting(Stage):Button
buildNewProject(Stage):Button
buildWelcomeScreen(Stage):FlowPane
buildStopImage():ImageView
buildTopMenu(Logger):MenuBar
volumeHandle(Slider):void
recordHandle():void
buildRecordFrame():VBox
alertHandle():void
buildBotToolbar(Stage):ToolBar
playBtn():Button
saveBtn():Button
buildImageView():ImageView
editHandle(Stage,Button,TextField):void
notValidInput(TextField):boolean
emptyFieldHandle():void
imageHandle(String):MediaPlayer
dragOverHandle(DragEvent):void
dropHandle(DragEvent,AudioButton):void
isImage(String):boolean
isAudio(String):boolean
buildAutomaticButton(AudioButton):void
buildInitialGui(Stage,Pagination):void
createPage(int):VBox
<<Java Class>>
TBCLog
log
primaryStage: Stage
logFile: File
listOfLines: ArrayList<String>
listOfLabels: ArrayList<Label>
logPane: VBox
label: Label
openlog: Button
showstats: Button
scrollpane: ScrollPane
TBCLog()
main(String[]):void
start(Stage):void
selectLoggerFile():void
removeLabel():void
readFile(File):void
createListView():void
showStats():void
+buttons
0..*
<<Java Class>>
AudioButton
config
serialVersionUID: long
name: String
AudioPath: String
ImagePath: String
AudioButton()
AudioButton(String,String,String)
getName():String
getAudioPath():String
getImagePath():String
setName(String):void
setAudioPath(String):void
setImagePath(String):void
chooseAudioPath():void
chooseImagePath():void
<<Java Interface>>
TalkBoxConfiguration
config
getNumberOfAudioButtons():int
getNumberOfAudioSets():int
getTotalNumberOfButtons():int
getRelativePathToAudioFiles():Path
getAudioFileNames():String[][]
<<Java Class>>
WebDownloader
config
audioPath: String
ImagePath: String
WebDownloader()
downloadAutomatic(String):void
downloadBestImage(String):void
getImages(String):List<String>
bestMatch(List<String>,String):String
DownloadImage(String,String):void
downloadTTS(String):void
<<Java Class>>
AudioRecord
config
stopped: boolean
recording: boolean
filepath: String
AudioRecord()
main(String[]):void
buildRecordImage():ImageView
buildStopImage():ImageView
buildRunButton():ImageView
runHandle():void
buildButtons():HBox
recordAudio():void
handleButton():void
alertHandle():void
recordAudio(String):void
buildFlowPane(Stage):FlowPane
start(Stage):void
<<Java Class>>
Stat
log
TestCounter: int
SaveCounter: int
RecordAudioCounter: int
AddNewCounter: int
DeleteLastCounter: int
DeleteSpecificCounter: int
AutofillAudioButtonCounter: int
DragDropAudioButtonCounter: int
TitlesSet: int
ImagesSelected: int
AudioTracksSelected: int
AudioButtonsTested: int
AudioButtonsTestedStop: int
AudioButtonsEdited: int
AudioButtonsPlayed: int
AudioButtonsStopped: int
VolumeCounter: int
UserManualCounter: int
ReportIssueCounter: int
ContactUsCounter: int
Stat()
getStatWindow():VBox
addStat(VBox,String,int):void
<<Java Class>>
Utilities
config
text: String
AudioPath: String
ImagePath: String
defaultPath: String
numSeconds: int
recordedAudioPath: String
fileName: String
resourcePath: String
logPath: String
Utilities()
fileChoose(Stage,boolean):String
chooseFile(Stage):File
chooseLoggerFile(Stage):File
configFileChoose(Stage):File
configFileSave(Stage):File
recordFileSave(Stage):File
setEditPrompt(Stage):GridPane
setGridPaneConstraints(Button,Button,Label,TextField,TextField):void
getText():String
getNumSeconds():int
getFileName():String

Sequence diagram

The sequence diagram highlights the common use cases and how objects that make up the TalkBox software interact with each other during run-time. The previous class diagram gave a more general view of the application but the following diagram gives a verbose illustration of how the application works under the hood.

Furthermore, it also captures how the life span of an object starts and how it interacts with the other objects depending on the conditional cases the user goes through, based on inputs. This specific sequence highlights the actions the user would take that would help him set up the talkbox application and save a file to be used later in the simulator or in the logger.

A higher quality picture could be found at <https://i.imgur.com/Nxrrf0W.png>.


User
TalkBoxApp Class
<<UI>>
Builder Class
AudioButton Class
TalkBoxApp Class
TBCLog Class
Launches JAR file
Create new TalkBox
Start();
AudioButton b = new AudioButton()
buildInitialGui(Stage primaryStage,Pagination p)
Creates default filler log
Logger logr = Logger.getLogger(
Builder.class.getName());
UI for for config
Alt
Auto-complete
Set button up using autocomplete
buildAutomaticButton(AudioButton b);
b.setImagePath(File a);
b.setAduioPath(File b);
b.setName(String input);
ImageView iv1 = buildImageView();
iv1.setImage(b.getImage());
Returns a filled out AudioButton object
logr.fine("AutoBt:....");
Utilities.AutoBtAudio
ButtonCounter++;
Image now clickable for sound
Edit
Create new button via edit
editHandle(Stage x, Button submit,
TextField textField);
Utilities.setEditPrompt(Stage);
Prompt to set fields
Set fields
SubmitHandle(int k, AudioButton currentButton,
ImageView iv1,TextField textField,Button edit,
Stage x)
currentButton.setImagePath(File a);
currentButton.setAduioPath(File b);
currentButton.setName(String input);
logr.fine("Button:....");
Utilities.....++;
Returns a filled out AudioButton object
ImageView iv1 = buildImageView();
iv1.setImage(currentButton.getImage());
Image now clickable for sound
Drag Drop
Drag drop image and sound
dropHandle(DragEvent a, AudioButton b);
b.setImagePath(File a);
b.setAduioPath(File b);
b.setName(String input);
Returns a filled out AudioButton object
ImageView iv1 = buildImageView();
iv1.setImage(currentButton.getImage());
logr.fine("Drag Drop:....");
Utilities.....++;
Image now clickable for sound
Alt
Regular Audio Playing
Clicking on button to play audio
playSound(int j, ImageView stop, ImageView iv1,
Logger log, Boolean choice);
ImageHandle(String
audioPath);
logr.fine("Played:....");
Utilities.....++;
setOnPlaying()
Plays audio
Stopping audio
Stopping audio early
playSound(int j, ImageView stop, ImageView iv1,
Logger log, Boolean choice);
ImageHandle(String audioPath);
stop.setOnMouseClicked();
logr.fine("Stop:....");
Utilities.....++;
Returns the image from buildStopImage()
Stops audio
Record button
recordHandle();
Prompt for file name and
save location
buildRecordFrame();
logr.fine("Record:....");
Utilities.....++;
Audio recording
Audio recording stop
Saves audio recording
Help menu item
buildTopMenu(Logger log);
MenuItem.setOnAction();
logr.fine("Help:....");
Utilities.....++;
Opens link in browser
Add button
addButtonHandle(Stage primaryStage);
buttons.add(null)
logr.fine("Add:....");
Utilities.....++;
buildInitialGui(primaryStage, pagination());
Unset audio button
Delete last button
deleteLastButtonHandle(Stage
primaryStage);
buttons.remove(buttons.size() - 1);
logr.fine("Delete:....");
Utilities.....++;
buildInitialGui(primaryStage, pagination());
Remove last button
Delete specific button
deleteSpecificButtonHandle(Stage
primaryStage);
buttons.remove(i);
logr.fine("Delete:....");
Utilities.....++;
buildInitialGui(primaryStage, pagination());
Remove button i
Saving configuration
saveBtn();
Utilities.configFileSave(new Stage());
Prompt
for save location and file name
Alt
Invalid Input
Invalid file name or cancel save
filename == null
logr.fine("Save button: fail....");
Utilities.....++;
Do not create file
Configuration screen
Valid input
Valid save name
filename == input from user
Default filler log copied over into new
log file based filname and second log
file made for simulation logging.
logr.fine("Save button:....");
Utilities.....++;
saveSerializedFile();
Configuration screen
Open Simulator
playBtn();
TalkBoxApp a = new
TalkBoxApp(openSerializedFile(file));
logr.fine("TEST:....");
Utilities.....++;
Starts logging actions
into simulator log file
a.buildApplication(new Stage());
Opens simulator in new window
Clicking on audio buttons
Using load builder
Playing or stopping audio passing
in simulator logger
Opening TBCLogger
MenuBar logToolBar =
builder.buildTopMenu(logr);
TBCLog sa = new TBCLog();
TBCLogger in new window
sa.start(new Stage());
Opening stats
showStats();
Scene sc = new Scene
(Stat.getStatWindow());
Open stats in smaller window
Opening logs
selectLoggerFile();
Alt
Sim logs
Opening simulator logs
Config Logs
Opening config logs
Loads out log file
createListView();
Closing software
Delete default log file