Use Case Model

Class Hub

Release: 1.0
Version: 1.2
Date: 12/01/17
Responsible Analyst: Matthew Del Fante
E-mail: matthew.delfante@oit.edu

Signatory Page

Professor:

Signature (Calvin Caldwell)

Date

Analyst:

Matthew Del Fante

10/18/17

Signature (Matthew Del Fante)

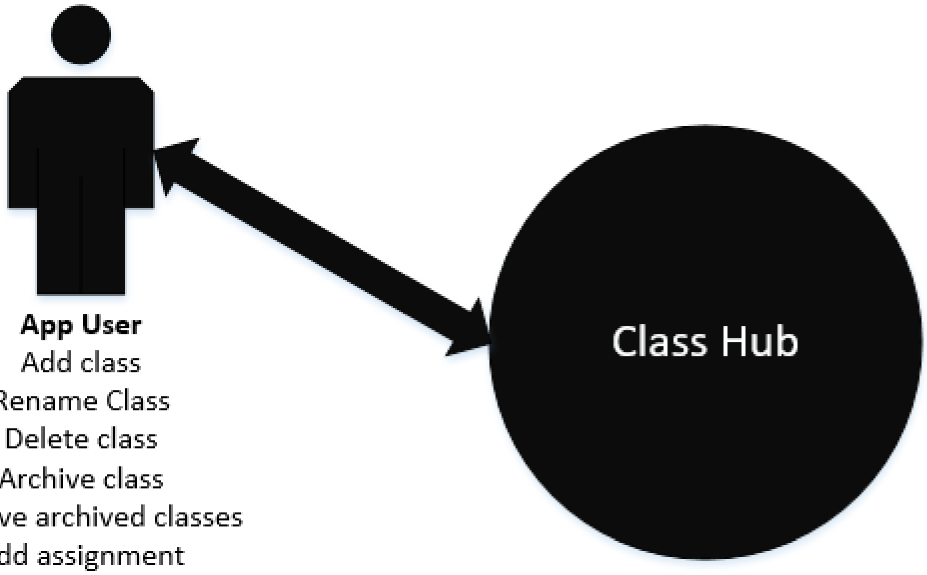
Date

Revision History

Author	Company	Version	Date	Filename	Comments
Matthew Del Fante	Delware	1.0	10/18/17	UseCaseModel.doc	Initial draft.
Matthew Del Fante	Delware	1.1	11/06/17	UseCaseModel.doc	Updated CRUD Matrix.
Matthew Del Fante	Delware	1.2	12/01/17	UseCaseModel.doc	Integrated dynamic model.

Contents

Context Diagram



App User
Add class
Rename Class
Delete class
Archive class
Retrieve archived classes
Add assignment
Edit assignment information
Delete assignment
Mark assignment as completed
View assignment information
Add audio recording
Rename audio recording
Delete audio recoding
Listen to audio recording
Add video recording
Rename video recording
Delete video recording
Watch video recording
Add notes
Edit notes
Rename notes
Delete notes
View notes
View assignments in assignment calendar
Tap assignment in assignment calendar

Use Case Catalog

Use Case Catalog ID	Use Case Name	Description
UC 01	Add Assignment 01	User adds an assignment to a class.
UC 02	Add Audio Recording 01	User adds an audio recording to a class.
UC 03	Add Class 01	User adds a class to the application.
UC 04	Add Notes 01	User adds a note to a class.
UC 05	Add Video Recording 01	User adds a video recording to a class.
UC 06	Archive Class 01	User archives a class.
UC 07	Delete Assignment 01	User deletes an assignment from a class.
UC 08	Delete Audio Recording 01	User deletes an audio recording from a class.
UC 09	Delete Class 01	User deletes a class from the application.
UC 10	Delete Note 01	User deletes a note from a class.
UC 11	Delete Video Recording 01	User deletes a video recording from a class.
UC 12	Edit Assignment 01	User edits information about an assignment.
UC 13	Edit Notes 01	User edits a note.
UC 14	Listen to Audio 01	User listens to an audio recording.
UC 15	MAAC 01	User marks an assignment as completed.
UC 16	Rename Audio Recording 01	User changes the name of an audio recording.
UC 17	Rename Class 01	User changes the name of a class.
UC 18	Rename Note 01	User changes the name of a note.
UC 19	Rename Video Recording 01	User changes the name of a video recording.
UC 20	Retrieve Archived Classes 01	Archived classes are reinserted into the application.
UC 21	View Assignment 01	User views an assignment’s information.
UC 22	View Notes 01	User views notes.
UC 23	Watch Video Recording 01	User watches a video recording.
UC 24	VAIAC 01	User views assignments in the Assignment Calendar.
UC 25	TAIAC 01	User taps an assignment in the Assignment Calendar.

Actor Catalog

Name	Type	Description
App User	Person	Someone who uses the Class Hub app to organize class information.

Features Verification Matrix

Feature Numbers	Use Case Catalog ID	Use Case Name
3.4.1, 3.2.1.3, 4.1	UC 01	Add Assignment 01
4.3, 6.1	UC 02	Add Audio Recording 01
3.3 - 3.3.1.1.1.2.2	UC 03	Add Class 01
4.5, 9.1	UC 04	Add Notes 01
4.4, 8	UC 05	Add Video Recording 01
3.4.1.2, 3.4.1.2.1, & 3.4.1.2.1.3.	UC 06	Archive Class 01
4.2, 5.1.1.2, 3.2.1.3	UC 07	Delete Assignment 01
6.2, 7.1 & 7.1.3	UC 08	Delete Audio Recording 01
3.4.1.2 & 3.4.1.2.1, 3.4.1.2.1.2	UC 09	Delete Class 01
9.2, 10.1 & 10.1.3	UC 10	Delete Note 01
8	UC 11	Delete Video Recording 01
5.1 & 5.1.1, 4.2	UC 12	Edit Assignment 01
9.2, 10.1 & 10.1.1	UC 13	Edit Notes 01
7.1 & 7.1.1	UC 14	Listen to Audio 01
4.2, 5.1.1.1	UC 15	MAAC 01
6.2, 7.1 & 7.1.2	UC 16	Rename Audio Recording 01
3.4.1.2 & 3.4.1.2.1, 3.4.1.2.1.1, 3.4.1.2.1.4	UC 17	Rename Class 01
9.2, 10.1 & 10.1.2	UC 18	Rename Note 01
8	UC 19	Rename Video Recording 01
3.3.1.2	UC 20	Retrieve Archived Classes 01
4.2., 5.1	UC 21	View Assignment 01
9.2, 10.1.1	UC 22	View Notes 01
8	UC 23	Watch Video Recording 01
3.2.1.1, 3.2.1.2, 3.2.1.3	UC 24	VAIAC 01
3.2.1.4	UC 25	TOAIAC 01

Use Case Specifications

Use Case Summary

In the Use Case Summary, I provide information that is the same for every use case.

Actors:
The actor of each use case is the App User. The App User is someone who uses the Class Hub app to organize class information.

Preconditions:
The preconditions for each use case are:

1. The app user must have an android device.
2. The app user must have the Class Hub app installed onto his/her android device.

Assumptions:
An assumption for each use case is that the actor can read English.

UC 01

General Information	
Use Case Name\Number: Add Assignment 01 Subject Area: Adding an assignment Description: The user wants to add a class assignment to a class.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
3.4.1, 3.4.1.1 & 3.4.1.1.1	The user navigates to the Class Activity of a specific class.
4.1	The process of adding an assignment.
3.2.1.2	Updating the Assignment Calendar when the assignment is added.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/13/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus

The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps

Number	Description	Adds/Alt Name/Number	Bus Rule#
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that he/she wants to add an assignment to.		
3	The system redirects the user to the Class Activity.		
4	The user presses the add assignment button.		
5	The system displays a pop up menu as described in requirement # 4.1.		
6	The user enters an assignment name, due date/time, priority level and notes.		
7	The user presses the done button.		
8	The system saves the assignment to the application.		
9	The system updates the Assignment Calendar to include the assignment.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user presses the cancel button.	The system makes the pop up menu disappear and redirects the user to the Class Activity.	
The user doesn’t enter an assignment’s name and/or due date/time .	The system keeps the done button on the pop up menu greyed out and unclickable.	
The user enters a non-unique assignment name.	The system will display a message saying that the assignment names must be unique per class.	

Post-Conditions

#	Description
1	The system saves the assignment to the application.
2	The system updates the Assignment Calendar to include the assignment.
3	The system makes the pop up menu disappear.
4	The system redirects the user to the Class Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ClassActivity	Handles the logic behind the UI of the Class Activity.	List<string> assignments

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/13/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency:	Minimum: 1		Maximum:		Average: 4	(OR)Fixed:
Per:	Hour: □	Day: □	Week: ☒	Month: □	Other:	

Timing Information

#	At / Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	at	9	milliseconds			2	The Assignment Calendar on the Home Activity should be updated with the new assignment instantaneously.

Volume Information

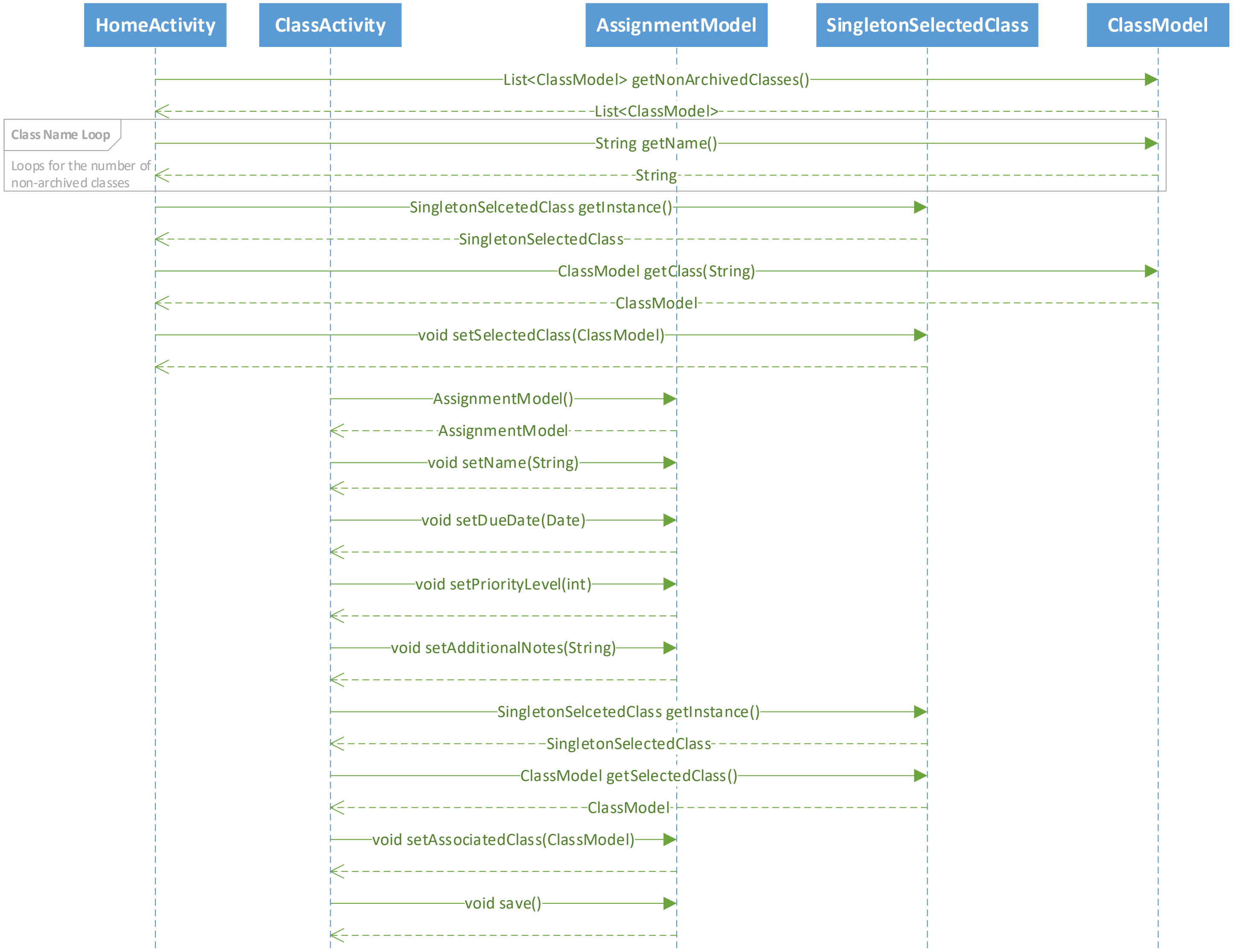
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 01 Add Assignment Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that he/she wants to add an assignment to.
    - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
    - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user presses the add assignment button.
    - The ClassActivity constructs an AssignmentModel object.
5. The system displays a pop up menu as described in requirement # 4.1.
6. The user enters an assignment name, due date/time, priority level and notes.
7. The user presses the done button.
    - The ClassActivity calls the AssignmentModel's setName(String), setDueDate(Date), setPriorityLevel(int) and setAdditionalNotes(String) methods and passes the corresponding user inputted values as parameters to those methods.
    - The ClassActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method.
    - The ClassActivity passes the return value of the getSelectedClass() method as a parameter to the AssignmentModel's setAssociatedClass(ClassModel) method.
8. The system saves the assignment to the application.
    - The ClassActivity calls the AssignmentModel's save() method (This is an ActiveAndroid method) saving the AssignmentModel to the database.
9. The system updates the Assignment Calendar to include the assignment.

UC 01 Add Assignment Sequence Diagram 01


HomeActivity
ClassActivity
AssignmentModel
SingletonSelectedClass
ClassModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
AssignmentModel()
AssignmentModel
void setName(String)
void setDueDate(Date)
void setPriorityLevel(int)
void setAdditionalNotes(String)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
void setAssociatedClass(ClassModel)
void save()

UC 02

General Information	
Use Case Name\Number: Add Audio Recording 01 Subject Area: Adding an audio recording. Description: The user wants to add an audio recording to a class.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
4.3	Directs the user to the Audio Recordings Activity.
6 & 6.1	Allows the user to add an audio recording.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that he/she wants to add an audio recording to.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Audio Recordings button.		
5	The system redirects the user to the Audio Recordings Activity.		
6	The user presses the button to start an audio recording.		
7	The system starts recording audio.	See Alt 01 (Pause/Continue Recording Audio)	
8	After some time, the user presses the Stop Audio Recording button.		
9	The system displays two buttons, one to delete the audio recording and one to save the audio recording.		
10	The user presses the button to save the audio recording.		
11	The system saves the audio recording.		

Exception Conditions

Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
The user presses the button to delete the audio recording.	The system will not save the audio recording.	
The user tries to save an audio recording that would cause the android device's non-volatile memory capacity to be exceeded.	The system will display a message saying the audio recording size is too large and not save the audio recording.	

Post-Conditions

#	**Description**
1	The system saves the audio recording to the application (the default name is the current time and date in military time).
2	The system displays the Audio Recordings Activity.

Candidate Objects

Class/Object Name	**Descriptions**	**Possible attributes**
AudioRecordingsActivity	Handles the logic behind the UI of the Audio Recordings Activity.	MediaRecorder mRecorder

Assumptions

#	**Assumption**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
1	The user added a class to the app.	10/15/17	Matthew Del Fante		

Issues

#	**Issue**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
1	What if the user denies Class Hub permission to use the microphone?	10/15/17	Matthew Del Fante		

Other Comments

Author	**Comment**	**Date**
Matthew Del Fante	If the user denies Class Hub microphone permission, then the user won't be able to record audio with the app.	10/15

Frequency of Execution

Frequency:	Minimum: 0		Maximum:	Average: 1	(OR)Fixed:
Per:	Hour: □	Day: □	Week: ☒	Month: □	Other:

Timing Information

#	**At/ Between**	**Step(s)**	**Timing Unit**	**Minimum**	**Average**	**Maximum**	**Comments**
N/A							

Volume Information

#	**Step #**	**Unit of Measure**	**Minimum**	**Average**	**Maximum**	**Comments**
1	11	Bytes			Size of android device's non-volatile memory.	The user can't save an audio recording that will cause the device's non-volatile memory to be exceeded.

Alternate Course General Information

Alternate Course Name\Number: Alt 01 (Pause/Continue Recording Audio)
Description: When recording audio, the user wants to be able to stop recording temporarily and then continue recording audio later.
Reason for Execution: The user may want to pause an audio recording and continue recording audio at a later time without creating two separate audio files.
Non Exception: ☒ Exception: □

Insertion Point

Step Inserted After
7

Pre-Conditions

1.	The user presses the button to start an audio recording.
2.	The system starts recording audio.

Alternate Course Steps

#	Step Description	Adds Use Case #	Business Rule(s) #
1	After some time, the user presses the pause button.		
2	The system stops recording audio.		
3	The system turns the pause button into a continue button.		
4	After some time the user presses the continue button.		
5	The system begins recording audio again.		
6	The system turns the continue button into a pause button.		

Post-Conditions

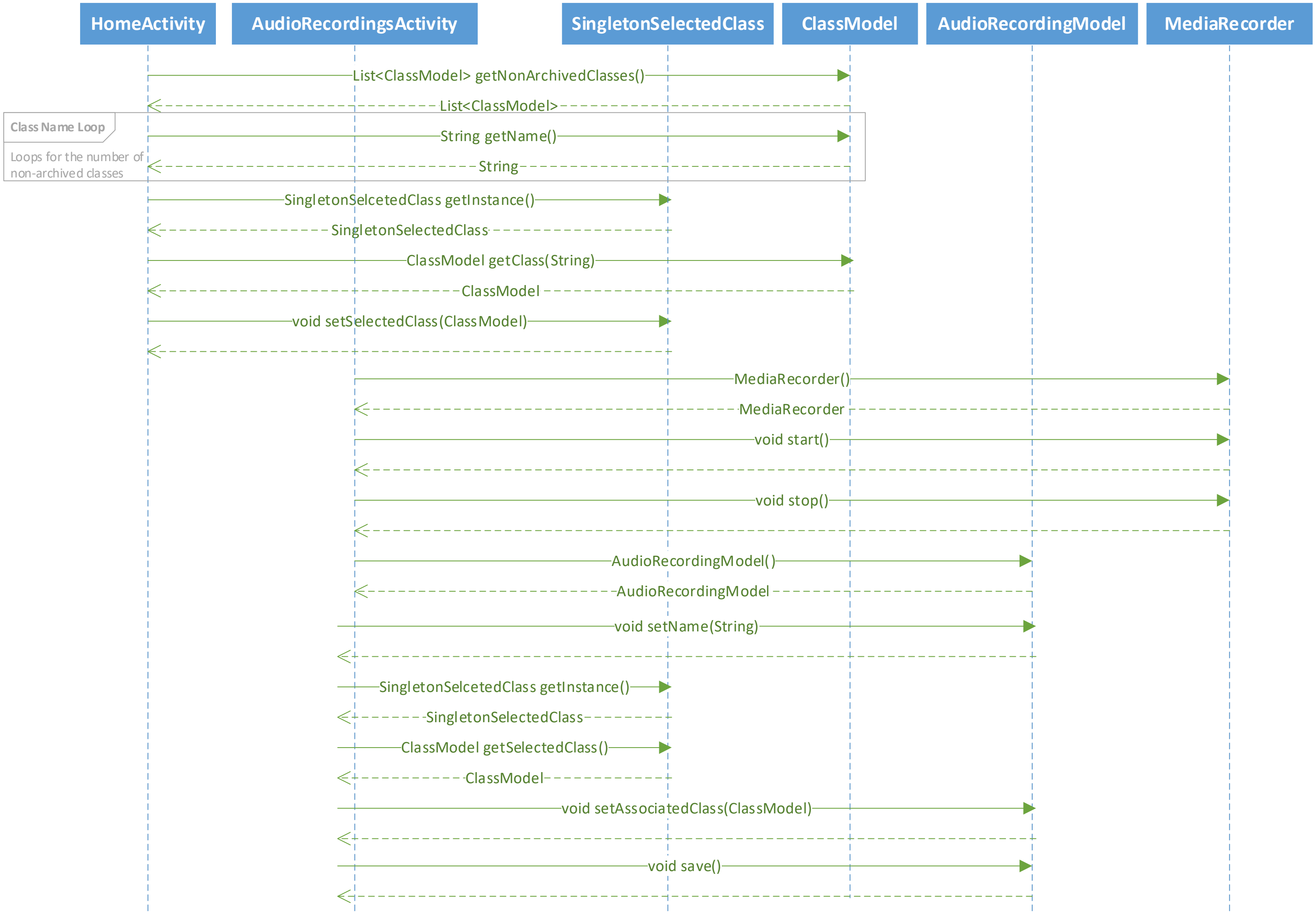
1	The system allows the user the ability to temporarily stop recoding audio and continue recording audio later.
2	The user can repeat this process as many times as he/she would like.

UC 02 Add Audio Recording Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel’s getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel’s getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes’ names.
2. The user taps the button of the class that he/she wants to add an audio recording to.
    - The HomeActivity calls the SingletonSelectedClass’ getInstance() method.
    - The HomeActivity calls the ClassModel’s getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass’ setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Audio Recordings button.
5. The system redirects the user to the Audio Recordings Activity.
6. The user presses the button to start an audio recording.
    - The AudioRecordingsActivity constructs a MediaRecorder object (Note: The MediaRecorder is an Android class).
7. The system starts recording audio.
    - The AudioRecordingsActivity calls the MediaRecorder’s start() method in order to start recording audio.
8. After some time, the user presses the Stop Audio Recording button.
    - The AudioRecordingsActivity calls the MediaRecorder’s stop() method in order to stop recording audio.
9. The system displays two buttons, one to delete the audio recording and one to save the audio recording.
10. The user presses the button to save the audio recording.
11. The system saves the audio recording.
    - The AudioRecordingsActivity constructs an AudioRecordingModel object.
    - The AudioRecordingsActivity calls the AudioRecordingModel’s setName(String) method passing the current date and military time as a parameter to the method.
    - The AudioRecordingsActivity calls the SingletonSelectedClass’ getInstance() method followed by the SingletonSelectedClass’ getSelectedClass() method.
    - The AudioRecordingsActivity then passes the return value of the getSelectedClass() method as a parameter to the AudioRecordingModel’s setAssociatedClass(ClassModel) method.
    - The AudioRecordingsActivity calls the AudioRecordingModel’s save() method (This is an ActiveAndroid method) saving that AudioRecordingModel to the database.

UC 02 Add Audio Recording Sequence Diagram 01

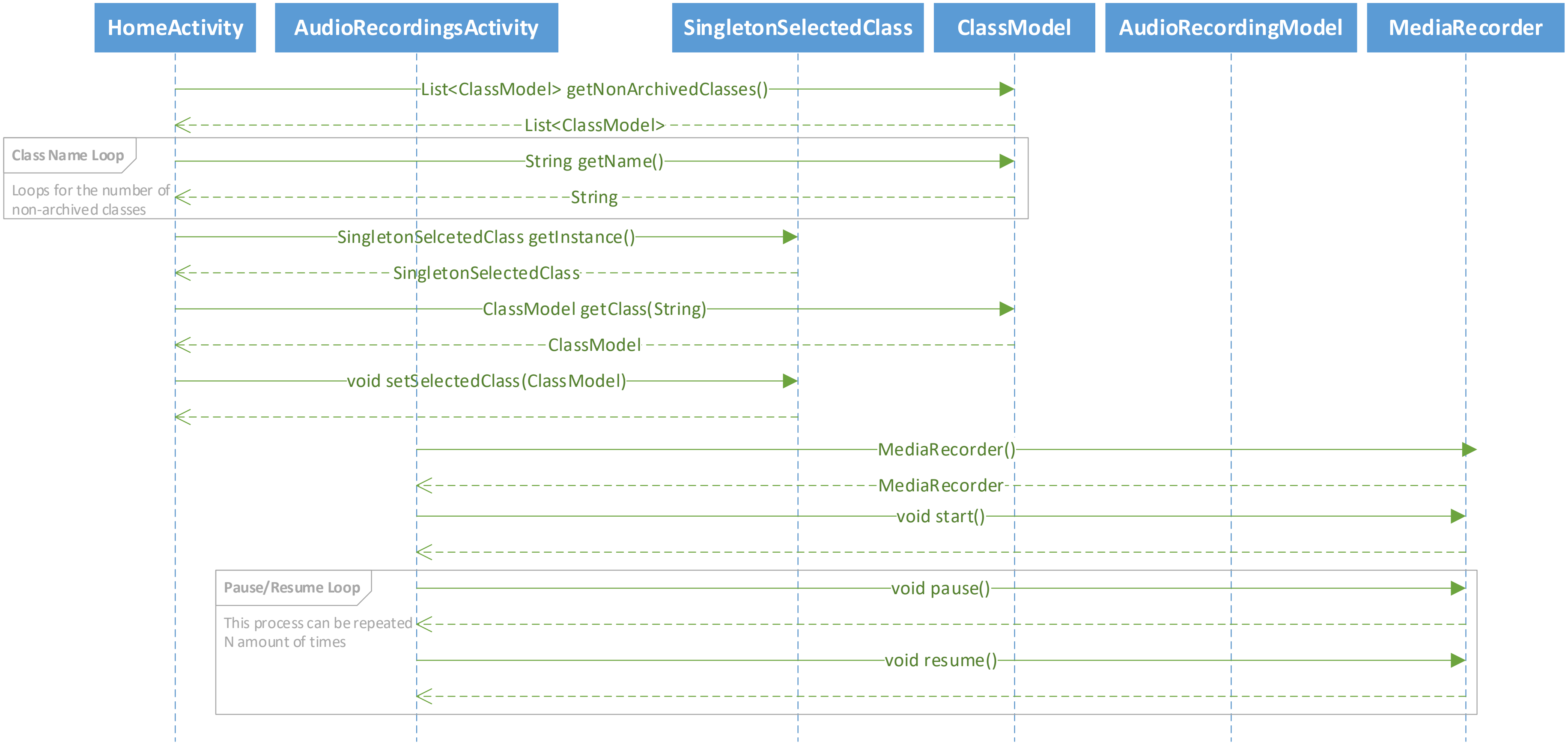


UC 02 Add Audio Recording Scenario 02

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
   - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
   - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
   - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that he/she wants to add an audio recording to.
   - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
   - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
   - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Audio Recordings button.
5. The system redirects the user to the Audio Recordings Activity.
6. The user presses the button to start an audio recording.
   - The AudioRecordingsActivity constructs a MediaRecorder object (Note: The MediaRecorder is an Android class).
7. The system starts recording audio.
   - The AudioRecordingsActivity calls the MediaRecorder's start() method in order to start recording audio.
8. After some time, the user presses the pause button.
9. The system stops recording audio.
   - The AudioRecordingsActivity calls the MediaRecorder's pause() method in order to pause the audio recording.
10. The system turns the pause button into a continue button.
11. After some time the user presses the continue button.
12. The system begins recording audio again.
    - The AudioRecordingsActivity calls the MediaRecorder's resume() method in order to continue the audio recording.
13. The system turns the continue button into a pause button.

UC 02 Add Audio Recording Sequence Diagram 02



UC 03

General Information	
Use Case Name\Number : Add Class 01 Subject Area : Adding a class Description: The user wants to add a class to the Class Hub application.	Responsible Analyst : Matthew Del Fante

Revision History

Author	Date	Comments
Matthew Del Fante	10/10/17	Initial draft.

Insertion Points in other Use Cases

Use Case Name	Use Case Number	Step Inserted After
N/A		

Actors

Actor Name	Person/System	Brief Description
See Use Case Summary.		

Pre-Conditions

#	Description
	See Use Case Summary.

Start Stimulus

The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps

Number	Description	Adds/Alt Name/Number	Bus Rule#
1	The system takes the user to the Class Hub Home Activity.		
2	The user presses the Add Class button on the Home Activity.		
3	The system displays a pop up menu as described in requirement # 3.3.1.1.1.		
4	The user types in a class name.		
5	The user taps the done button.		
6	The system adds the class to the application.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user taps the cancel button.	The system will make the pop up menu disappear.	
The user enters a non-unique class name.	The system will notify the user that he/she already has a class with that name.	

Post-Conditions

#	Description
1	The system adds the class to the application.
2	The pop up menu disappears.
3	The Home Activity will have a new button for the newly added class.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
HomeActivity	Handles the logic behind the UI of the Home Activity.	List<string> classNames

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
	See Use Case Summary.				

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency:	Minimum: 1	Maximum:	Average: 4	(OR)Fixed:	
Per:	Hour: □	Day: □	Week: □	Month: □	Other: School Term

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
N/A							

Volume Information

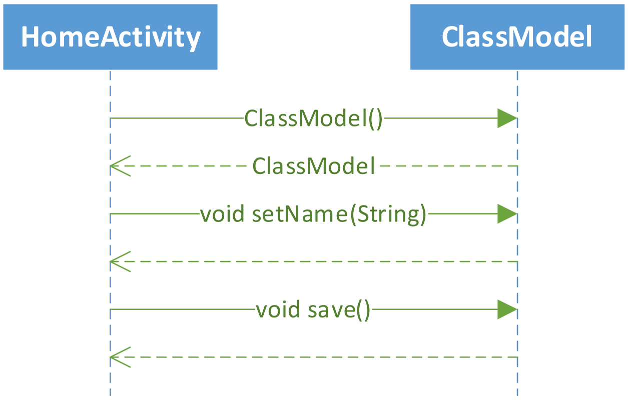
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 03 Add Class Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
2. The user presses the Add Class button on the Home Activity.
    - The HomeActivity constructs a ClassModel object.
3. The system displays a pop up menu as described in requirement # 3.3.1.1.1.
4. The user types in a class name.
    - The HomeActivity calls the ClassModel's setName(String) method and passes the user inputted name as a parameter to that method.
5. The user taps the done button.
6. The system adds the class to the application.
    - The HomeActivity calls the ClassModel's save() method (This is an ActiveAndroid method) saving the ClassModel to the database.

UC 03 Add Class Sequence Diagram 01



UC 04

General Information	
Use Case Name\Number: Add Notes 01 Subject Area: Adding Notes to a class. Description: The user wants to add notes to a class.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
4.5	Directs the user to the Notes Activity.
9.1	The process of adding a note to a class.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that he/she wants to add a note to.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Notes button.		
5	The system redirects the user to the Notes Activity.		
6	The user taps the button to start a note.		
7	The system displays a large textbox with save and cancel buttons.		
8	The user types some notes into the textbox.		
9	The user presses the save button.		
10	The system saves the note to the application (the name of the note defaults to the current time and date in military time).		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user tries to save a note that is only whitespace.	The save button will be greyed out and unclickable.	
The user presses the cancel button instead of the save button.	The note is discarded, the textbox disappears and the user is redirected to the Notes Activity.	

Post-Conditions

#	Description
1	The note is saved to the application.
2	The textbox disappears.
3	The user is redirected to the Notes Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
NotesActivity	Handles the logic behind the UI of the Notes Activity.	String mCurrentNote

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/15	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 1 (OR)Fixed:
Per: Hour: □ Day: □ Week: □ Month: □ Other: School term.

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
N/A							

Volume Information

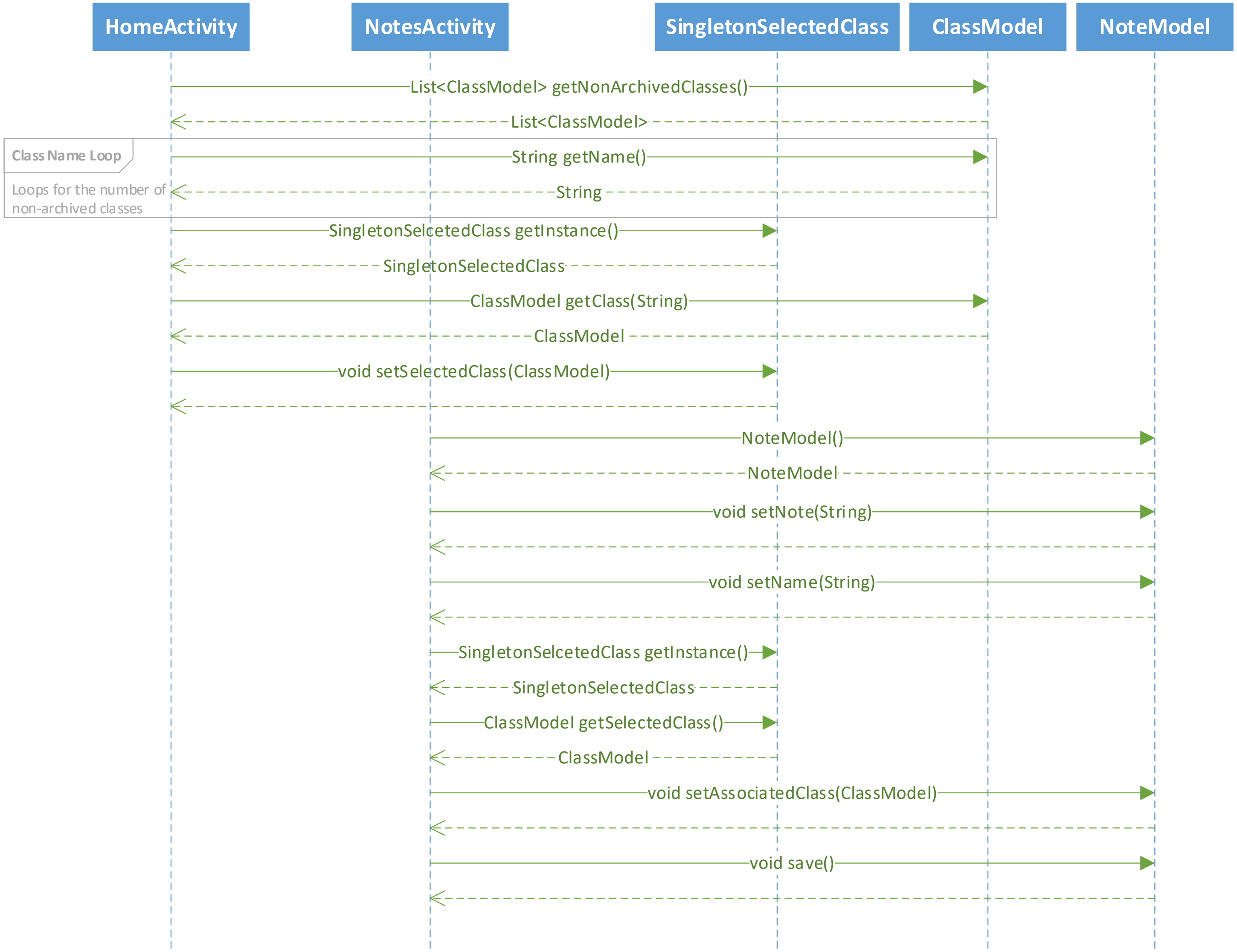
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
1	10	bytes			1 billion	Sqlite can’t store more than 1 billion bytes per string.

UC 04 Add Notes Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
   - The HomeActivity calls the ClassModel’s getNonArchivedClasses() method which returns a List<ClassModel>.
   - The HomeActivity then loops over the List<ClassModel> calling the ClassModel’s getName() method.
   - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes’ names.
2. The user taps the button of the class that he/she wants to add a note to.
   - The HomeActivity calls the SingletonSelectedClass’ getInstance() method.
   - The HomeActivity calls the ClassModel’s getClass(String) method passing the name of the class the user just tapped.
   - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass’ setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Notes button.
5. The system redirects the user to the Notes Activity.
6. The user taps the button to start a note.
7. The system displays a large textbox with save and cancel buttons.
8. The user types some notes into the textbox.
9. The user presses the save button.
   - The NotesActivity constructs a NoteModel object.
   - The NotesActivity calls the NoteModel’s setNote(String) method passing the user inputted notes as a parameter to that method.
10. The system saves the note to the application (the name of the note defaults to the current time and date in military time).
    - The NotesActivity calls the NoteModel’s setName(String) method passing the current date and military time as a parameter to the method.
    - The NotesActivity calls the SingletonSelectedClass’ getInstance() method followed by the SingletonSelectedClass’ getSelectedClass() method.
    - The NotesActivity then passes the return value of the getSelectedClass() method as a parameter to the NoteModel’s setAssociatedClass(ClassModel) method.
    - The NotesActivity calls the NoteModel’s save() method (This is an ActiveAndroid method) saving that NoteModel to the database.

UC 04 Add Notes Sequence Diagram 01


HomeActivity
NotesActivity
SingletonSelectedClass
ClassModel
NoteModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
String getName()
Loops for the number of non-archived classes
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
NoteModel()
NoteModel
void setNote(String)
void setName(String)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
void setAssociatedClass(ClassModel)
void save()

UC 05

General Information	
Use Case Name\Number: Add Video Recording 01 Subject Area: Adding a video recording Description: The user wants to add a video recording to a class.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
4.4	Directs the user to the Video Recording Activity
8	Encompasses everything that has to do with videos in Class Hub.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that he/she wants to add a video recording to.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Video Recordings button.		
5	The system redirects the user to the Video Recordings Activity.		
6	The user presses the button to start a video recording.		
7	The system starts recording a video.	See Alt 01 (Pause/Continue Recording Video)	
8	After some time, the user presses the Stop Video Recording button.		
9	The system displays two buttons, one to delete the video recording and one to save the video recording.		
10	The user presses the button to save the video recording.		
11	The system saves the video recording.		

Exception Conditions

Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
The user presses the button to delete the video recording.	The system will not save the video recording.	
The user tries to save a video recording that would cause the android device's non-volatile memory capacity to be exceeded.	The system will display a message saying the video recording size is too large and not save the video recording.	

Post-Conditions

#	**Description**
1	The system saves the video recording to the application (the default name is the current time and date in military time).
2	The system displays the Video Recording Activity.

Candidate Objects

Class/Object Name	**Descriptions**	**Possible attributes**
VideoRecordingsActivity	Handles the logic behind the UI of the Video Recordings Activity.	MediaRecorder mRecorder, Camera mCamera

Assumptions

#	**Assumption**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
1	The user added a class to the app.	10/15/17	Matthew Del Fante		

Issues

#	**Issue**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
1	What if the user denies Class Hub permission to use the camera?	10/15/17	Matthew Del Fante		

Other Comments

Author	**Comment**	**Date**
Matthew Del Fante	If the user denies Class Hub camera permission, then the user won't be able to record video with the app.	10/15

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 1 (OR)Fixed:
Per: Hour: □ Day: □ Week: ☒ Month: □ Other:

Timing Information

#	**At/ Between**	**Step(s)**	**Timing Unit**	**Minimum**	**Average**	**Maximum**	**Comments**
N/A							

Volume Information

#	**Step #**	**Unit of Measure**	**Minimum**	**Average**	**Maximum**	**Comments**
1	11	Bytes			Size of android device's non-volatile memory.	The user can't save a video recording that will cause the device's non-volatile memory to be exceeded.

Alternate Course General Information

Alternate Course Name\Number: Alt 01 (Pause/Continue Recording Video)
Description: When recording video, the user wants to be able to stop recording temporarily and then continue recording video.
Reason for Execution: The user may want to pause a video recording and continue recording video later without creating two separate video files.
Non Exception: ☒ Exception: □

Insertion Point

Step Inserted After
7

Pre-Conditions

1.	The user presses the button to start a video recording.
2.	The system starts recording video.

Alternate Course Steps

#	**Step Description**	**Adds Use Case #**	**Business Rule(s) #**
1	After some time, the user presses the pause button.		
2	The system stops recording video.		
3	The system turns the pause button into a continue button.		
4	After some time the user presses the continue button.		
5	The system begins recording video again.		
6	The system turns the continue button into a pause button.		

Post-Conditions

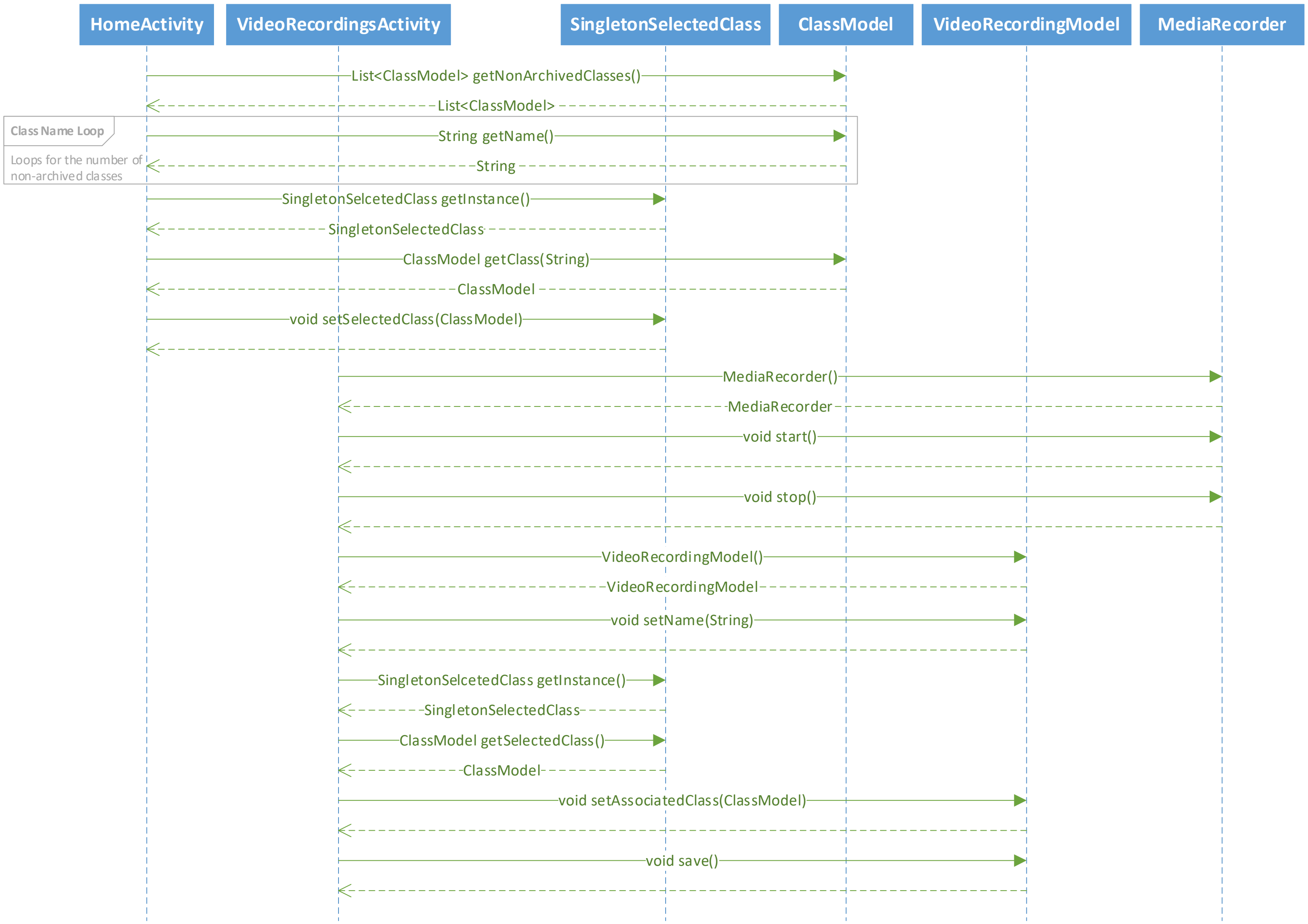
1	The system allows the user the ability to temporarily stop recoding video and continue recording video later.
2	The user can repeat this process as many times as he/she would like.

UC 05 Add Video Recording Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that he/she wants to add a video recording to.
    - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
    - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Video Recordings button.
5. The system redirects the user to the Video Recordings Activity.
6. The user presses the button to start a video recording.
    - The VideoRecordingsActivity constructs a MediaRecorder object (Note: The MediaRecorder is an Android class).
7. The system starts recording a video.
    - The VideoRecordingsActivity calls the MediaRecorder's start() method in order to start recording video.
8. After some time, the user presses the Stop Video Recording button.
    - The VideoRecordingsActivity calls the MediaRecorder's stop() method in order to stop recording video.
9. The system displays two buttons, one to delete the video recording and one to save the video recording.
10. The user presses the button to save the video recording.
11. The system saves the video recording.
    - The VideoRecordingsActivity constructs a VideoRecordingModel object.
    - The VideoRecordingsActivity calls the VideoRecordingModel's setName(String) method passing the current date and military time as a parameter to the method.
    - The VideoRecordingsActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method
    - Then the VideoRecordingsActivity passes the return value of the getSelectedClass() method as a parameter to the VideoRecordingModel's setAssociatedClass(ClassModel) method.
    - The VideoRecordingsActivity calls the VideoRecordingModel's save() method (This is an ActiveAndroid method) saving that VideoRecordingModel to the database.

UC 05 Add Video Recording Sequence Diagram 01

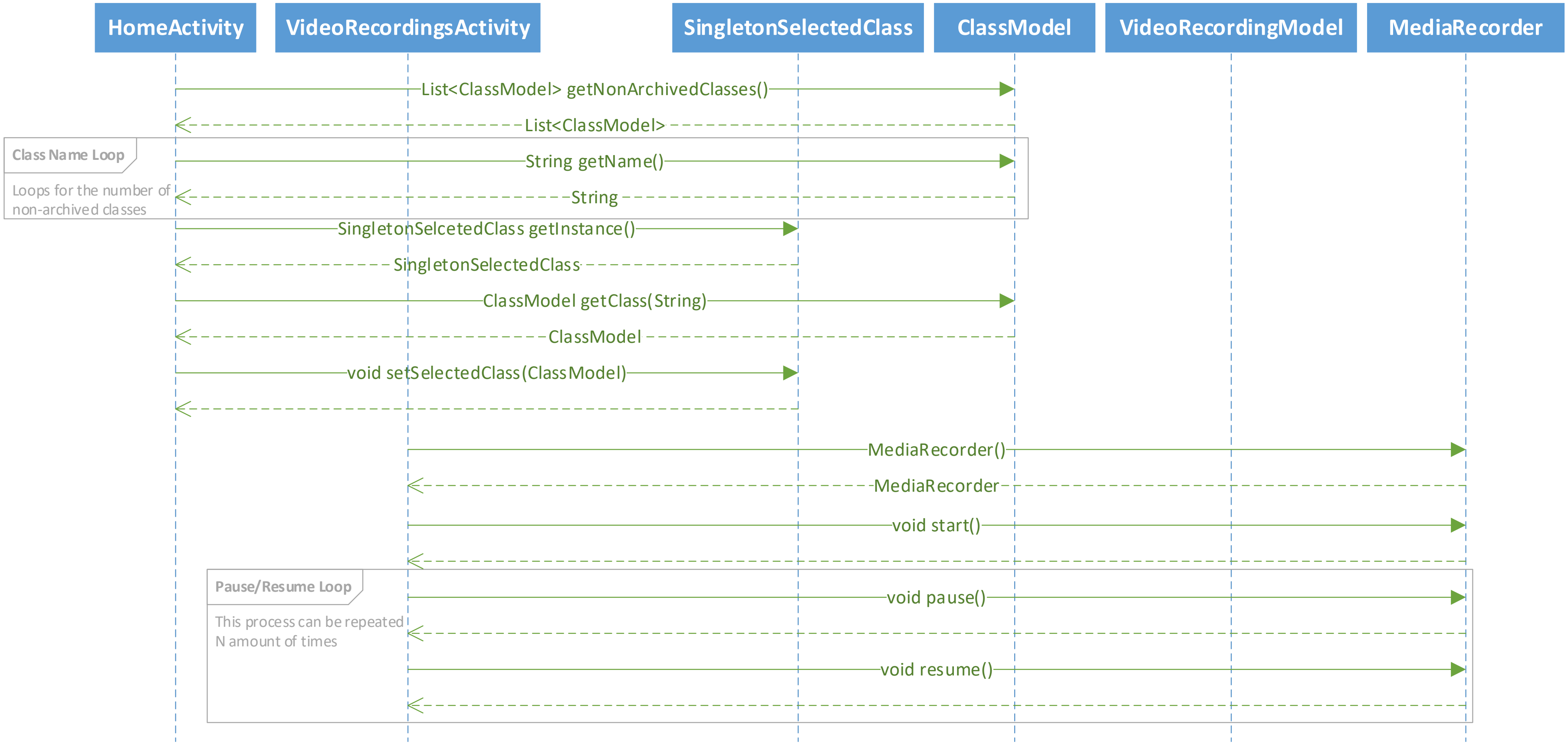


UC 05 Add Video Recording Scenario 02

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel’s getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel’s getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes’ names.
2. The user taps the button of the class that he/she wants to add a video recording to.
    - The HomeActivity calls the SingletonSelectedClass’ getInstance() method.
    - The HomeActivity calls the ClassModel’s getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass’ setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Video Recordings button.
5. The system redirects the user to the Video Recordings Activity.
6. The user presses the button to start a video recording.
    - The VideoRecordingsActivity constructs a MediaRecorder object (Note: The MediaRecorder is an Android class).
7. The system starts recording a video.
    - The VideoRecordingsActivity calls the MediaRecorder’s start() method in order to start recording video.
8. After some time, the user presses the pause button.
9. The system stops recording video.
    - The VideoRecordingsActivity calls the MediaRecorder’s pause() method in order to pause the video recording.
10. The system turns the pause button into a continue button.
11. After some time the user presses the continue button.
12. The system begins recording video again.
    - The VideoRecordingsActivity calls the MediaRecorder’s resume() method in order to continue the video recording.
13. The system turns the continue button into a pause button.

UC 05 Add Video Recording Sequence Diagram 02



UC 06

General Information	
Use Case Name\Number: Archive Class 01 Subject Area: Archiving a class Description: The app user wants to archive a class he/she already added to the Class Hub app.	Responsible Analyst : Matthew Del Fante

Requirements/Feature Trace

REQ#	Requirements Name and / or Short Description
3.4.1.2 & 3.4.1.2.1	The user long presses a class to display a pop up menu
3.4.1.2.1.3.	The user chooses to archive a class.

Revision History

Author	Date	Comments
Matthew Del Fante	10/13/17	Initial draft.

Insertion Points in other Use Cases

Use Case Name	Use Case Number	Step Inserted After
N/A		

Actors

Actor Name	Person/System	Brief Description
See Use Case Summary.		

Pre-Conditions

#	Description
	See Use Case Summary.

Start Stimulus

The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps

Number	Description	Adds/Alt Name/Number	Bus Rule#
1	The system takes the user to the Class Hub Home Activity.		
2	The user long presses on the name of a class.		
3	The system displays a pop up menu as described in requirement # 3.4.1.2.1.		
4	The user taps the archive class button.		
5	The system saves the class's information to the database.		
6	The system removes all instances of the class from the UI.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
N/A		

Post-Conditions

#	Description
1	The system saves the class's information to the database.

2	The system removes all instances of the class from the UI.
3	The system redirects the user to the Home Activity.

Candidate Objects

Class/Object Name	**Descriptions**	**Possible attributes**
HomeActivity	Handles the logic behind the UI of the Home Activity.	List<string> classNames

Assumptions

#	**Assumption**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
1	The user added a class to the app.	10/13/17	Matthew Del Fante		

Issues

#	**Issue**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
N/A					

Other Comments

Author	**Comment**	**Date**
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 4 (OR)Fixed:

Per: Hour: □ Day: □ Week: □ Month: □ Other: School Term

Timing Information

#	**At/ Between**	**Step(s)**	**Timing Unit**	**Minimum**	**Average**	**Maximum**	**Comments**
1	at	6	milliseconds			1	Removing UI instances of the class should be instantaneous.

Volume Information

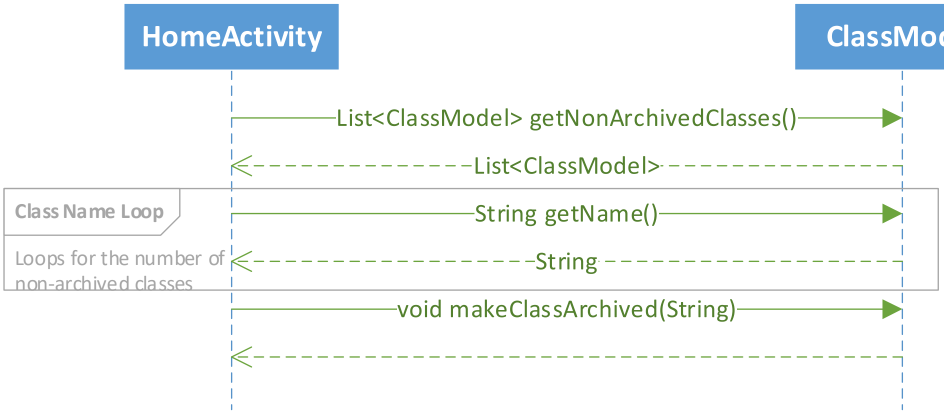
#	**Step #**	**Unit of Measure**	**Minimum**	**Average**	**Maximum**	**Comments**
N/A						

UC 06 Archive Class Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
   - The HomeActivity calls the ClassModel’s getNonArchivedClasses() method which returns a List<ClassModel>.
   - The HomeActivity then loops over the List<ClassModel> calling the ClassModel’s getName() method.
   - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes’ names.
2. The user long presses on the name of a class.
3. The system displays a pop up menu as described in requirement # 3.4.1.2.1.
4. The user taps the archive class button.
5. The system saves the class’s information to the database.
   - The HomeActivity calls the ClassModel’s makeClassArchived(String) method passing the name of the class the user long pressed in step 2 as a parameter to the method.
6. The system removes all instances of the class from the UI.

UC 06 Archive Class Sequence Diagram 01


HomeActivity
ClassModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
void makeClassArchived(String)

UC 07

General Information	
Use Case Name\Number: Delete Assignment 01 Subject Area: Deleting an assignment from a class. Description: A user wants to delete an assignment from a class.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
4.2	Navigates the user to the View Assignments Activity
5.1.1.2	Allows a user to delete an assignment from the class hub app.
3.2.1.2	Updating the Assignment Calendar when the assignment is deleted.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/14/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that he/she wants to delete an assignment from.		
3	The system redirects the user to the Class Activity.		
4	The user taps the View Assignments button.		
5	The system redirects the user to the View Assignments Activity.		
6	The user taps the assignment he/she wants to edit.		
7	The system displays a pop up menu as described in requirement # 5.1.1.		
8	The user taps the Delete Assignment button.		
9	The system displays a confirmation message asking if the user is sure he/she wants to delete the assignment.		
10	The user taps the yes button.		
11	The system deletes the assignment from the application.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user doesn't confirm he or she wants to delete the assignment.	The system does not delete the assignment from the application and the pop up menu in step 7 is displayed.	

Post-Conditions

#	Description
1	The system deletes the assignment from the application.
2	The system deletes the assignment from the Assignment Calendar.
3	The system makes the pop up menu disappear.
4	The system redirects the user to the View Assignments Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ViewAssignmentsActivity	Handles the logic behind the UI of the View Assignments Activity.	List<string> assignments

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/13/17	Matthew Del Fante		
2	The user added an assignment to the app.	10/13/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency:	Minimum: 0	Maximum:	Average: 4	(OR)Fixed:	
Per:	Hour: □	Day: □	Week: ☒	Month: □	Other:

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	at	11	millisecond			1	Deleting the assignment should be instantaneous with respect to the UI and Assignment Calendar.

Volume Information

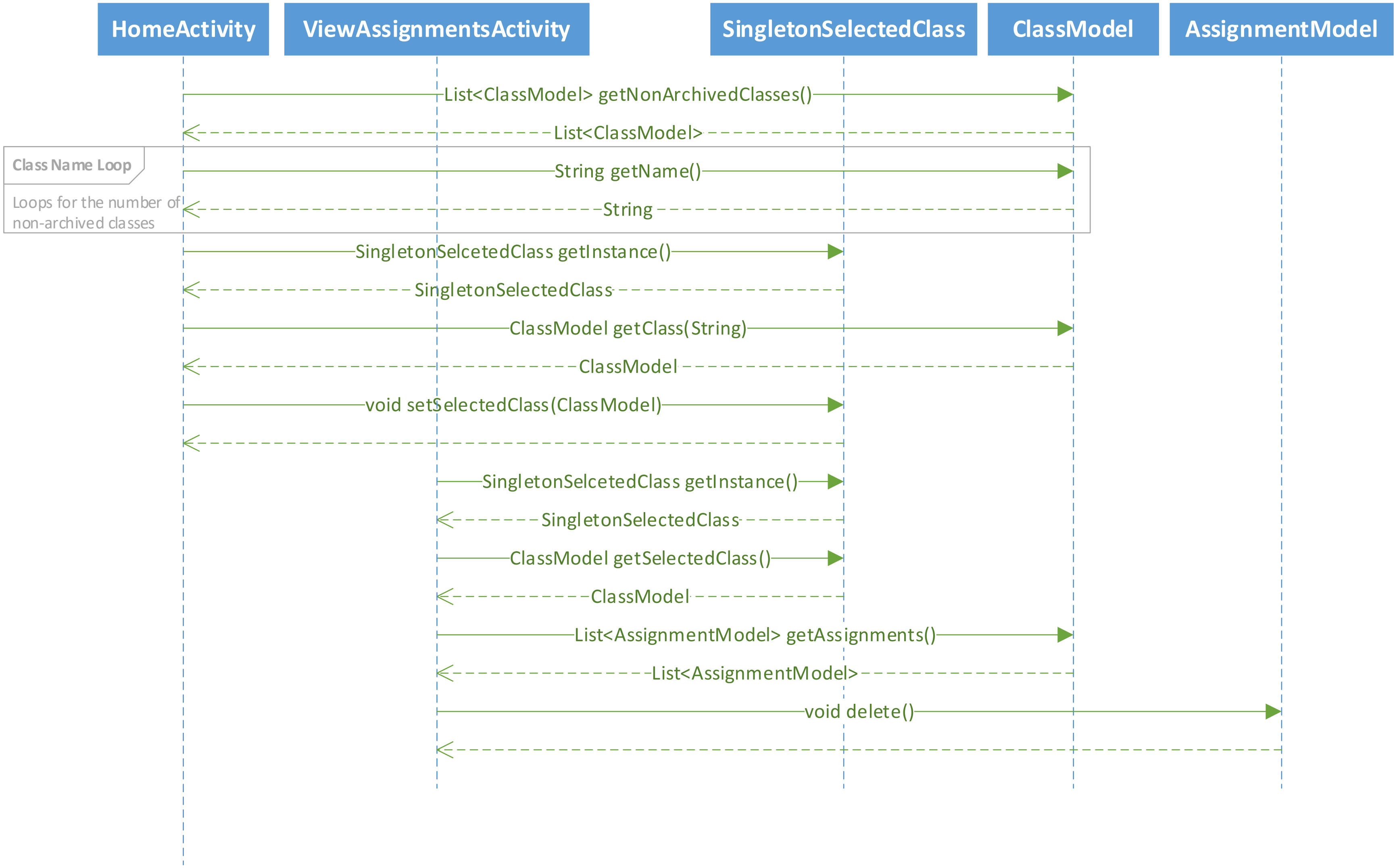
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 07 Delete Assignment Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that he/she wants to delete an assignment from.
    - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
    - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the View Assignments button.
5. The system redirects the user to the View Assignments Activity.
    - The ViewAssignmentsActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method. The ViewAssignmentsActivity then uses the return value of the getSelectedClass() method to call the ClassModel's getAssignments() method.
    - The List<AssignmentModel> that the getAssignments() method returns represents all the assignments that show up in the View Assignments Activity.
6. The user taps the assignment he/she wants to delete.
7. The system displays a pop up menu as described in requirement # 5.1.1.
8. The user taps the Delete Assignment button.
9. The system displays a confirmation message asking if the user is sure he/she wants to delete the assignment.
10. The user taps the yes button.
11. The system deletes the assignment from the application.
    - The ViewAssignmentsActivity calls the AssignmentModel's delete() method (this is an ActiveAndroid method) using the AssignmentModel instance the user selected on step 6. This deletes the AssignmentModel from the database.

UC 07 Delete Assignment Sequence Diagram 01


HomeActivity
ViewAssignmentsActivity
SingletonSelectedClass
ClassModel
AssignmentModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<AssignmentModel> getAssignments()
List<AssignmentModel>
void delete()

UC 08

General Information

Use Case Name\Number: Delete Audio Recording 01 Subject Area: Deleting an audio recording. Description: The user wants to delete an audio recording.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace

REQ#	Requirements Name and / or Short Description
6.2	Redirects the user to the View Audio Recordings Activity.
7.1 & 7.1.3	Allows the user to delete an audio recording.

Revision History

Author	Date	Comments
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases

Use Case Name	Use Case Number	Step Inserted After
N/A		

Actors

Actor Name	Person/System	Brief Description
See Use Case Summary.		

Pre-Conditions

#	Description
	See Use Case Summary.

Start Stimulus

The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps

Number	Description	Adds/Alt Name/Number	Bus Rule#
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that he/she wants to listen to delete an audio recording from.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Audio Recordings button.		
5	The system redirects the user to the Audio Recordings Activity.		
6	The user taps the View Audio Recordings button.		
7	The system redirects the user to the View Audio Recordings Activity.		
8	The user taps the audio recording he/she would like to delete.		
9	The system displays a pop up menu as described in requirement # 7.1.		
10	The user presses the button to delete an audio recording.		
11	The system displays a confirmation message asking if the user is sure he/she wants to delete the audio recording.		
12	The user taps the yes button.		

13	The system deletes the audio recording from the application.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user doesn't confirm he/she wants to delete the audio recording.	The audio recording isn't deleted from the application and the pop up menu in step 9 is displayed.	

Post-Conditions

#	Description
1	The system deletes the audio recording from the application.
2	The system makes the pop up menu disappear.
3	The system redirects the user to the View Audio Recordings Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ViewAudioRecordingsActivity	Handles the logic behind the UI of the View Audio Recordings Activity.	List<string> audioNames

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/15/17	Matthew Del Fante		
2	The user added an audio recording to the app.	10/15/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency:	Minimum: 0		Maximum:	Average: 4	(OR)Fixed:
Per:	Hour: □	Day: □	Week: □	Month: ☒	Other:

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	at	13	millisecond			1	Deleting the audio recording should be instantons with respect to the UI.

Volume Information

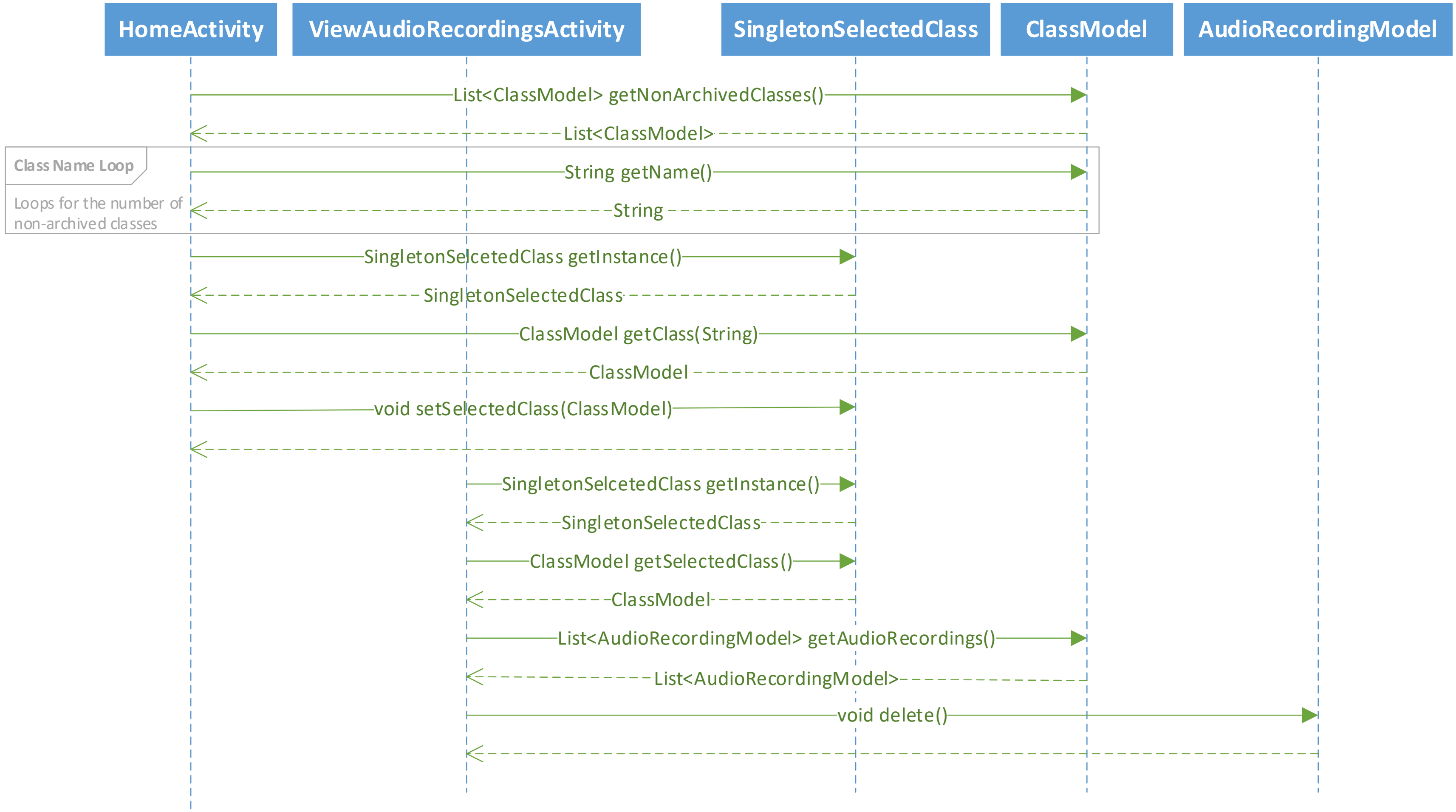
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 08 Delete Audio Recording Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that he/she wants to delete an audio recording from.
    - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
    - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Audio Recordings button.
5. The system redirects the user to the Audio Recordings Activity.
6. The user taps the View Audio Recordings button.
7. The system redirects the user to the View Audio Recordings Activity.
    - The ViewAudioRecordingsActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method. The ViewAudioRecordingsActivity then uses the return value of the getSelectedClass() method to call the ClassModel's getAudioRecordings() method.
    - The List<AudioRecordingModel> that the getAudioRecordings () method returns represents all the audio recordings that show up in the View Audio Recordings Activity.
8. The user taps the audio recording he/she would like to delete.
9. The system displays a pop up menu as described in requirement # 7.1.
10. The user presses the button to delete an audio recording.
11. The system displays a confirmation message asking if the user is sure he/she wants to delete the audio recording.
12. The user taps the yes button.
13. The system deletes the audio recording from the application.
    - The ViewAudioRecordingsActivity calls the AudioRecordingModel's delete() method (this is an ActiveAndroid method) using the AudioRecordingModel instance the user selected on step 8. This deletes the AudioRecordingModel from the database.

UC 08 Delete Audio Recording Sequence Diagram 01


HomeActivity
ViewAudioRecordingsActivity
SingletonSelectedClass
ClassModel
AudioRecordingModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<AudioRecordingModel> getAudioRecordings()
List<AudioRecordingModel>
void delete()

UC 09

General Information	
Use Case Name\Number: Delete Class 01 Subject Area : Deleting a class Description : The user wants to delete a class from the Class Hub application.	Responsible Analyst : Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
3.4.1.2 & 3.4.1.2.1	The user long presses a class to display a pop up menu.
3.4.1.2.1.2	The app user deletes a class from the Class Hub application.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/13/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
		See Use Case Summary.

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user long presses on the name of a class.		
3	The system displays a pop up menu as described in requirement # 3.4.1.2.1.		
4	The user taps the delete class button.		
5	The system displays a confirmation message asking if the user is sure he/she wants to delete the class.		
6	The user taps the yes button.		
7	The system deletes the class from the application.		

Exception Conditions		
Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
The user doesn't confirm he/she wants to delete the class.	The class isn't deleted from the app and the pop up menu in step 3 is displayed.	

Post-Conditions

#	Description
1	The system deletes the class from the application.
2	The system makes the pop up menu disappear.
3	The system redirects the user to the Home Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
HomeActivity	Handles the logic behind the UI of the Home Activity.	List<string> classNames

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/13/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency:	Minimum: 0		Maximum:	Average: 4	(OR)Fixed:
Per:	Hour: □	Day: □	Week: □	Month: □	Other: School Term

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	After	7	millisecond			1	Deleting the class should be instantons with respect to the UI.

Volume Information

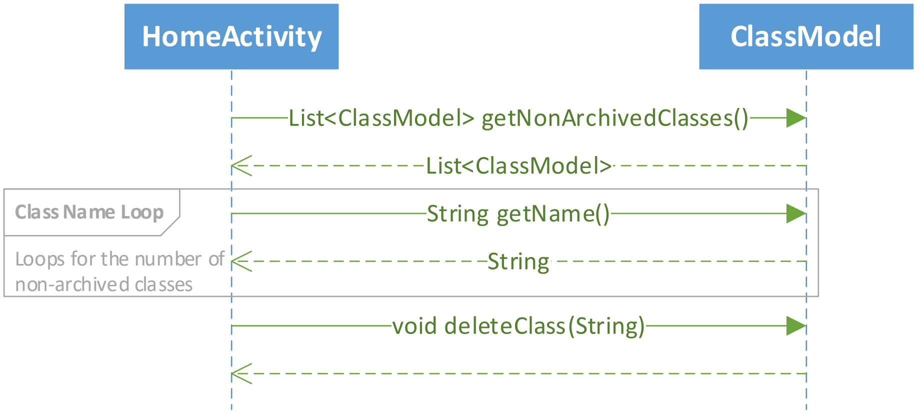
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 09 Delete Class Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
   - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
   - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
   - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user long presses on the name of a class.
3. The system displays a pop up menu as described in requirement # 3.4.1.2.1.
4. The user taps the delete class button.
5. The system displays a confirmation message asking if the user is sure he/she wants to delete the class.
6. The user taps the yes button.
7. The system deletes the class from the application.
   - The HomeActivity calls the ClassModel's deleteClass(String) method passing the name of the class the user long pressed in step 2 as a parameter to the method. The ClassModel with that name is then deleted from the database.

UC 09 Delete Class Sequence Diagram 01



UC 10

General Information	
Use Case Name\Number: Delete Note 01 Subject Area: Deleting a note. Description: The user wants to delete a note he/she saved.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace

REQ#	Requirements Name and / or Short Description
9.2	Redirects the user to the View Notes Activity
10.1 & 10.1.3	Allows a user to delete a note.

Revision History

Author	Date	Comments
Matthew Del Fante	10/15/17	Initial Draft.

Insertion Points in other Use Cases

Use Case Name	Use Case Number	Step Inserted After
N/A		

Actors

Actor Name	Person/System	Brief Description
See Use Case Summary.		

Pre-Conditions

#	Description
	See Use Case Summary.

Start Stimulus

The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps

Number	Description	Adds/Alt Name/Number	Bus Rule#
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that he/she wants to add a note to.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Notes button.		
5	The system redirects the user to the Notes Activity.		
6	The user taps the View All Notes button		
7	The system redirects the user to the View Notes Activity.		
8	The user taps on the note he/she would like to delete.		
9	The system displays a pop up menu as described in requirement # 10.1		
10	The user presses the Delete Note button.		
11	The system displays a confirmation message asking if the user is sure he/she wants to delete the note.		
12	The user taps the yes button.		
13	The system deletes the note from the application.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user doesn't confirm he/she wants to delete the note.	The note isn't deleted from the app and the pop up menu in step 9 is displayed.	

Post-Conditions

#	Description
1	The system deletes the note from the application.
2	The system makes the pop up menu disappear.
3	The system redirects the user to the View Notes Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ViewNotesActivity	Handles the logic behind the UI of the View Notes Activity.	List<string> mNotes

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/15	Matthew Del Fante		
2	The user added a note to a class.	10/15	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 4 (OR)Fixed:
Per: Hour: ☐ Day: ☐ Week: ☐ Month: ☐ Other: School term.

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	at	13	milliseconds			1	Deleting the note should be instantons with respect to the UI.

Volume Information

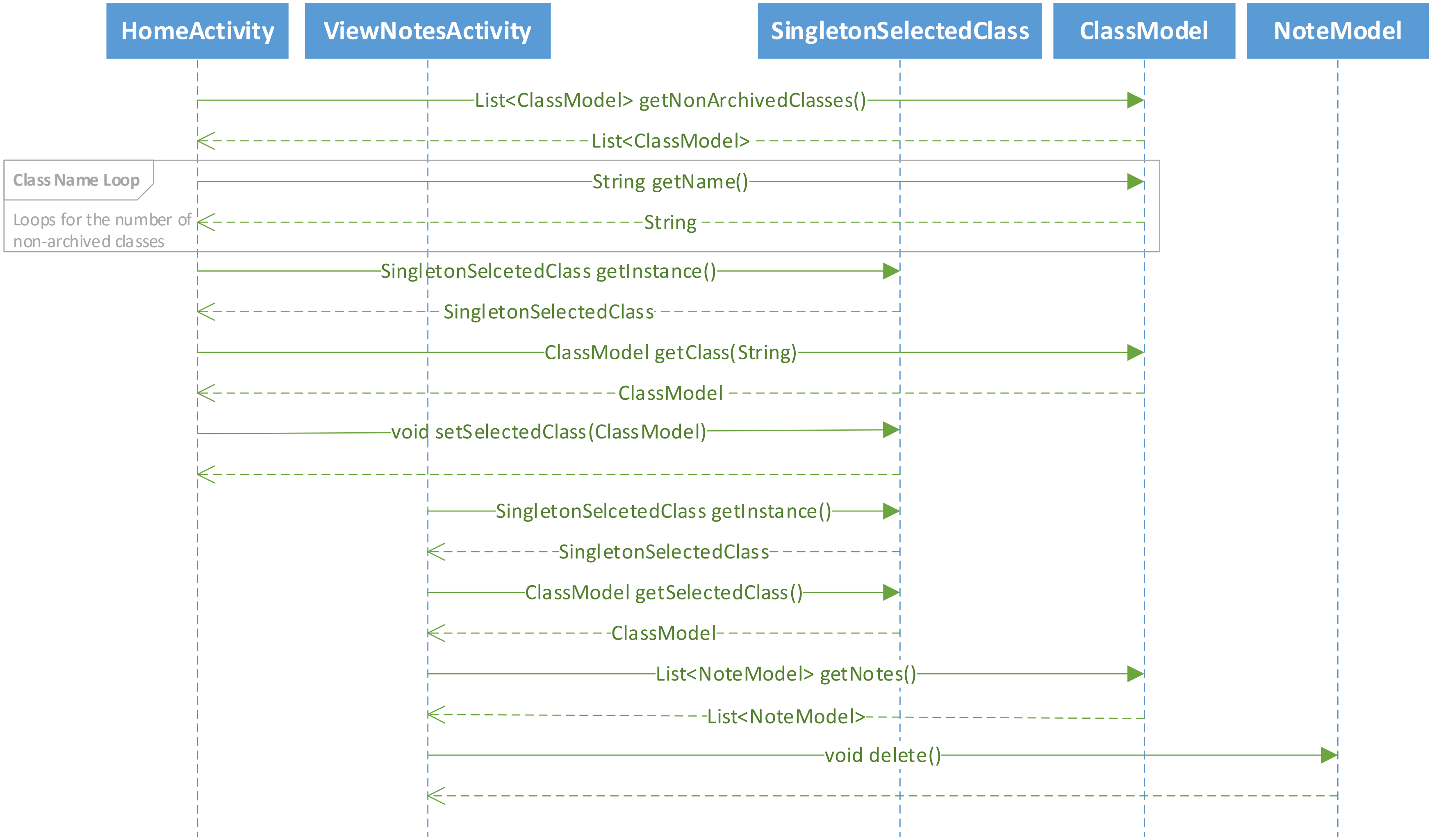
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 10 Delete Note Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
   - The HomeActivity calls the ClassModel’s getNonArchivedClasses() method which returns a List<ClassModel>.
   - The HomeActivity then loops over the List<ClassModel> calling the ClassModel’s getName() method.
   - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes’ names.
2. The user taps the button of the class that he/she wants to delete an assignment from.
   - The HomeActivity calls the SingletonSelectedClass’ getInstance() method.
   - The HomeActivity calls the ClassModel’s getClass(String) method passing the name of the class the user just tapped.
   - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass’ setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Notes button.
5. The system redirects the user to the Notes Activity.
6. The user taps the View All Notes button
7. The system redirects the user to the View Notes Activity.
   - The ViewNotesActivity calls the SingletonSelectedClass’ getInstance() method followed by the SingletonSelectedClass’ getSelectedClass() method. The ViewNotesActivity then uses the return value of the getSelectedClass() method to call the ClassModel’s getNotes() method.
   - The List<NoteModel> that the getNotes() method returns represents all the notes that show up in the View Notes Activity.
8. The user taps on the note he/she would like to delete.
9. The system displays a pop up menu as described in requirement # 10.1
10. The user presses the Delete Note button.
11. The system displays a confirmation message asking if the user is sure he/she wants to delete the note.
12. The user taps the yes button.
13. The system deletes the note from the application.
    - The ViewNotesActivity calls the NoteModel’s delete() method (this is an ActiveAndroid method) using the NoteModel instance the user selected on step 8. This deletes the NoteModel from the database.

UC 10 Delete Note Sequence Diagram 01


HomeActivity
ViewNotesActivity
SingletonSelectedClass
ClassModel
NoteModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<NoteModel> getNotes()
List<NoteModel>
void delete()

UC 11

General Information	
Use Case Name\Number: Delete Video Recording 01 Subject Area: Deleting a video recording. Description: The user wants to delete a video recording.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
8	Encompasses everything that has to do with videos in Class Hub.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that has the video the user wants to delete.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Video Recordings button.		
5	The system redirects the user to the Video Recordings Activity.		
6	The user taps the View Video Recordings button.		
7	The system redirects the user to the View Video Recordings Activity.		
8	The user taps the video recording he/she would like to delete.		
9	The system displays a pop up menu as described in requirement # 7.1, but with video information instead of audio information.		
10	The user presses the button to delete a video recording.		
11	The system displays a confirmation message asking if the user is sure he/she wants to delete the video recording.		
12	The user taps the yes button.		

13	The system deletes the video recording from the application.		

Exception Conditions

Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
The user doesn't confirm he/she wants to delete the video recording.	The video recording isn't deleted from the application and the pop up menu in step 9 is displayed.	

Post-Conditions

#	**Description**
1	The system deletes the video recording from the application.
2	The system makes the pop up menu disappear.
3	The system redirects the user to the View Video Recordings Activity.

Candidate Objects

Class/Object Name	**Descriptions**	**Possible attributes**
ViewVideoRecordingsActivity	Handles the logic behind the UI of the View Video Recordings Activity.	List<string> videoNames

Assumptions

#	**Assumption**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
1	The user added a class to the app.	10/15/17	Matthew Del Fante		
2	The user added an video recording to the app.	10/15/17	Matthew Del Fante		

Issues

#	**Issue**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
N/A					

Other Comments

Author	**Comment**	**Date**
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 4 (OR)Fixed:
Per: Hour: □ Day: □ Week: □ Month: ☒ Other:

Timing Information

#	**At/ Between**	**Step(s)**	**Timing Unit**	**Minimum**	**Average**	**Maximum**	**Comments**
1	at	13	millisecond			1	Deleting the video recording should be instantons with respect to the UI.

Volume Information

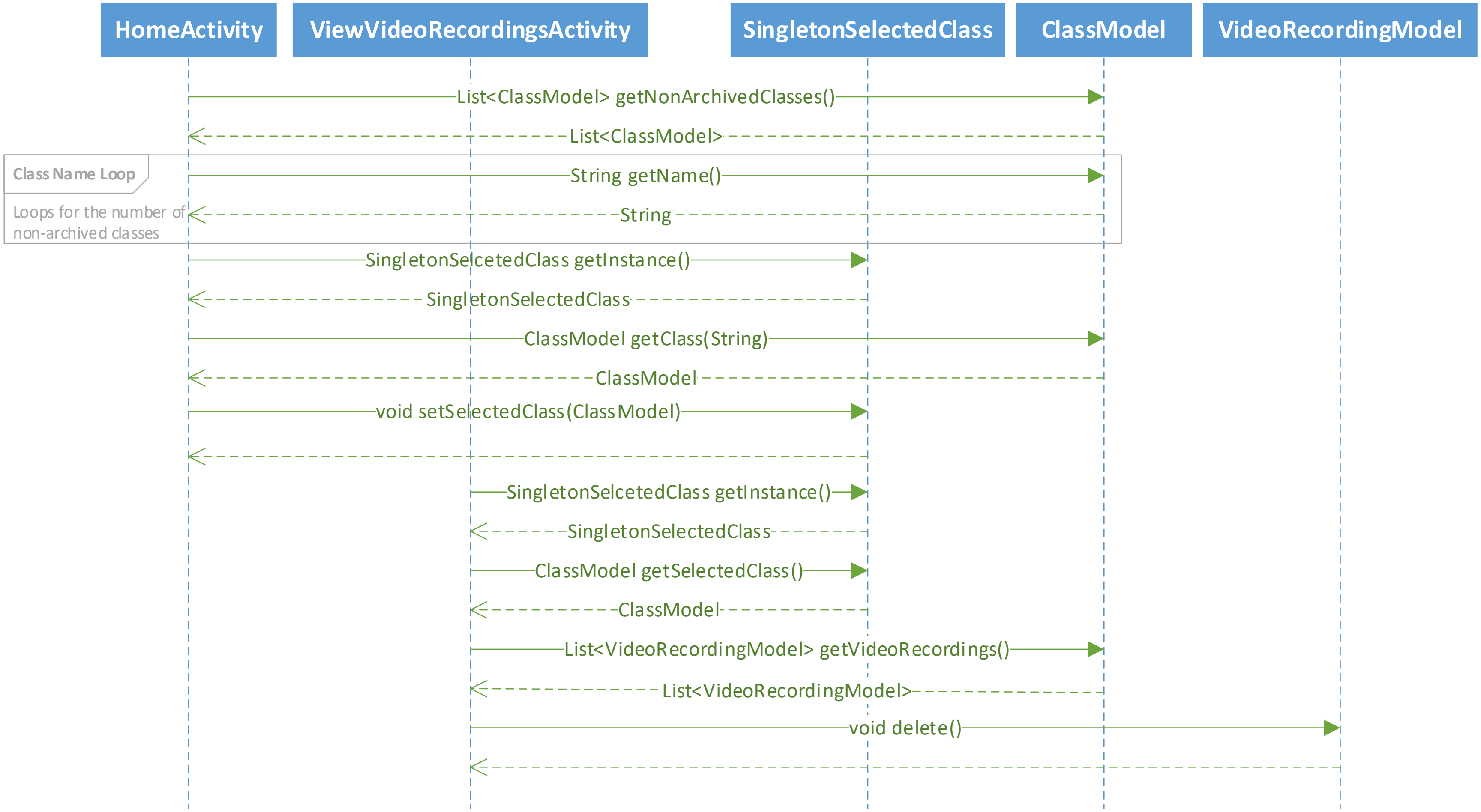
#	**Step #**	**Unit of Measure**	**Minimum**	**Average**	**Maximum**	**Comments**
N/A						

UC 11 Delete Video Recording Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel’s getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel’s getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes’ names.
2. The user taps the button of the class that has the video the user wants to delete.
    - The HomeActivity calls the SingletonSelectedClass’ getInstance() method.
    - The HomeActivity calls the ClassModel’s getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass’ setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Video Recordings button.
5. The system redirects the user to the Video Recordings Activity.
6. The user taps the View Video Recordings button.
7. The system redirects the user to the View Video Recordings Activity.
    - The ViewVideoRecordingsActivity calls the SingletonSelectedClass’ getInstance() method followed by the SingletonSelectedClass’ getSelectedClass() method. The ViewVideoRecordingsActivity then uses the return value of the getSelectedClass() method to call the ClassModel’s getVideoRecordings() method.
    - The List<VideoRecordingModel> that the getVideoRecordings() method returns represents all the video recordings that show up in the View Video Recordings Activity.
8. The user taps the video recording he/she would like to delete.
9. The system displays a pop up menu as described in requirement # 7.1, but with video information instead of audio information.
10. The user presses the button to delete a video recording.
11. The system displays a confirmation message asking if the user is sure he/she wants to delete the video recording.
12. The user taps the yes button.
13. The system deletes the video recording from the application.
    - The ViewVideoRecordingsActivity calls the VideoRecordingModel’s delete() method (this is an ActiveAndroid method) using the VideoRecordingModel instance the user selected on step 8. This deletes the VideoRecordingModel from the database.

UC 11 Delete Video Recording Sequence Diagram 01


HomeActivity
ViewVideoRecordingsActivity
SingletonSelectedClass
ClassModel
VideoRecordingModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<VideoRecordingModel> getVideoRecordings()
List<VideoRecordingModel>
void delete()

UC 12

General Information	
Use Case Name\Number: Edit Assignment 01 Subject Area: Editing an assignment's information. Description: A user wants to edit an assignment's information.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
4.2	Navigates the user to the View Assignments Activity
5.1 & 5.1.1	Allows a user to edit an assignment's information.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/13/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the class button that has the assignment he/she would like to edit.		
3	The system redirects the user to the Class Activity.		
4	The user taps the View Assignments button.		
5	The system redirects the user to the View Assignments Activity.		
6	The user taps the assignment he/she wants to edit.		
7	The system displays a pop up menu as described in requirement # 5.1.1.		
8	The user edits any of the assignment's information (name, due date/time, priority level, notes).		
9	The user presses the done button		
10	The system updates the assignment's information within the application.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user presses the cancel button.	The system makes the pop up menu disappear and redirects the user to the View Assignments Activity.	
The user deletes all characters out of the assignment name.	The system makes the done button on the pop up menu greyed out and unclickable.	
The user enters a non-unique assignment name.	The system will display a message saying that the assignment names must be unique per class.	

Post-Conditions

#	Description
1	The system updates the assignment's information within the application.
2	The system makes the pop up menu disappear.
3	The system redirects the user to the View Assignments Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ViewAssignmentsActivity	Handles the logic behind the UI of the View Assignments Activity.	List<string> assignments

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/13/17	Matthew Del Fante		
2	The user added an assignment to the app.	10/13/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency:	Minimum: 0		Maximum:	Average: 2	(OR)Fixed:
Per:	Hour: □	Day: □	Week: □	Month: ☒	Other:

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	at	10	milliseconds			1	The assignment's information should be updated instantaneously on the UI.

Volume Information

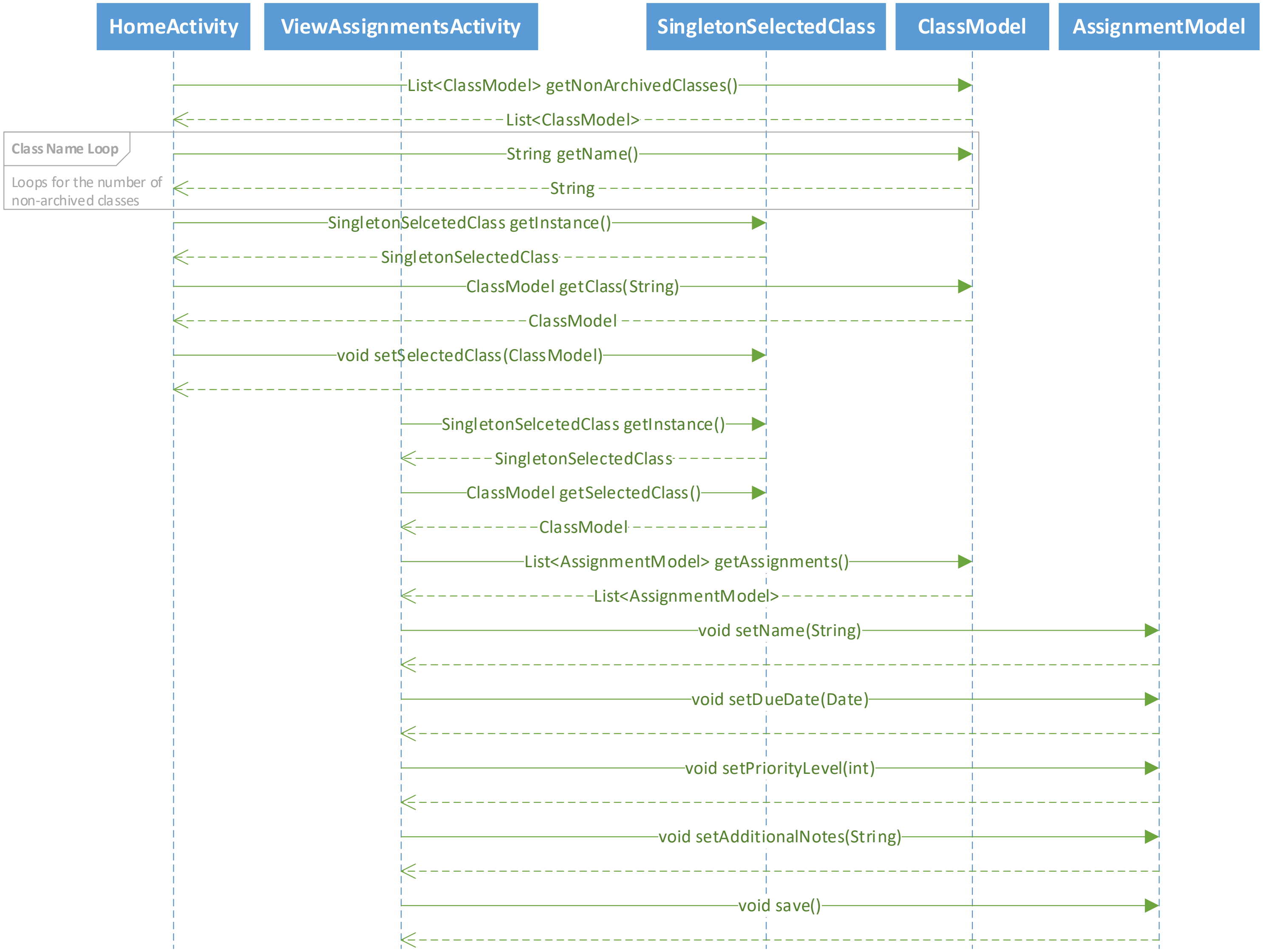
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 12 Edit Assignment Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that has the assignment he/she would like to edit.
    - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
    - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the View Assignments button.
5. The system redirects the user to the View Assignments Activity.
    - The ViewAssignmentsActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method. The ViewAssignmentsActivity then uses the return value of the getSelectedClass() method to call the ClassModel's getAssignments() method.
    - The List<AssignmentModel> that the getAssignments() method returns represents all the assignments that show up in the View Assignments Activity.
6. The user taps the assignment he/she wants to edit.
7. The system displays a pop up menu as described in requirement # 5.1.1.
8. The user edits any of the assignment's information (name, due date/time, priority level, notes).
9. The user presses the done button
    - The ViewAssignmentsActivity uses the AssignmentModel the user selected in step 6 to call its setName(String), setDueDate(Date), setPriorityLevel(int) and setAdditionalNotes(String) methods passing the corresponding user inputted values as parameters to those methods.
10. The system updates the assignment's information within the application.
    - The ViewAssignmentsActivity calls the AssignmentModel's save() method (This is an ActiveAndroid method) updating the AssignmentModel's values in the database.

UC 12 Edit Assignment Sequence Diagram 01


HomeActivity
ViewAssignmentsActivity
SingletonSelectedClass
ClassModel
AssignmentModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<AssignmentModel> getAssignments()
List<AssignmentModel>
void setName(String)
void setDueDate(Date)
void setPriorityLevel(int)
void setAdditionalNotes(String)
void save()

UC 13

General Information	
Use Case Name\Number: Edit Notes 01 Subject Area: Editing notes. Description: The user wants to edit a note about a class.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
9.2	Redirects the user to the View Notes Activity.
10.1 & 10.1.1	Displays the pop up menu that allows a user to edit a note.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/15	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that has the note he/she would like to edit.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Notes button.		
5	The system redirects the user to the Notes Activity.		
6	The user taps the View All Notes button		
7	The system redirects the user to the View Notes Activity.		
8	The user taps on the note he/she would like to edit.		
9	The system displays a pop up menu as described in requirement # 10.1		
10	The user presses the View/Edit Note button		
11	The system displays a large textbox with the contents of the note in it.		
12	The user edits the note.		
13	The user presses the Save Edits button.		
14	The system saves the edited note to the application.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user presses the Discard Edits button.	The edits the user made are discarded, the textbox disappears and the user is redirected to the View Notes Activity.	

Post-Conditions

#	Description
1	The system saves the edits to the note to the application.
2	The textbox disappears.
3	The user is redirected to the View Notes Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ViewNotesActivity	Handles the logic behind the UI of the View Notes Activity.	List<string> mNotes

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/15	Matthew Del Fante		
2	The user added a note to a class.	10/15	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 1 (OR)Fixed:

Per: Hour: □ Day: □ Week: □ Month: □ Other: School term.

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
N/A							

Volume Information

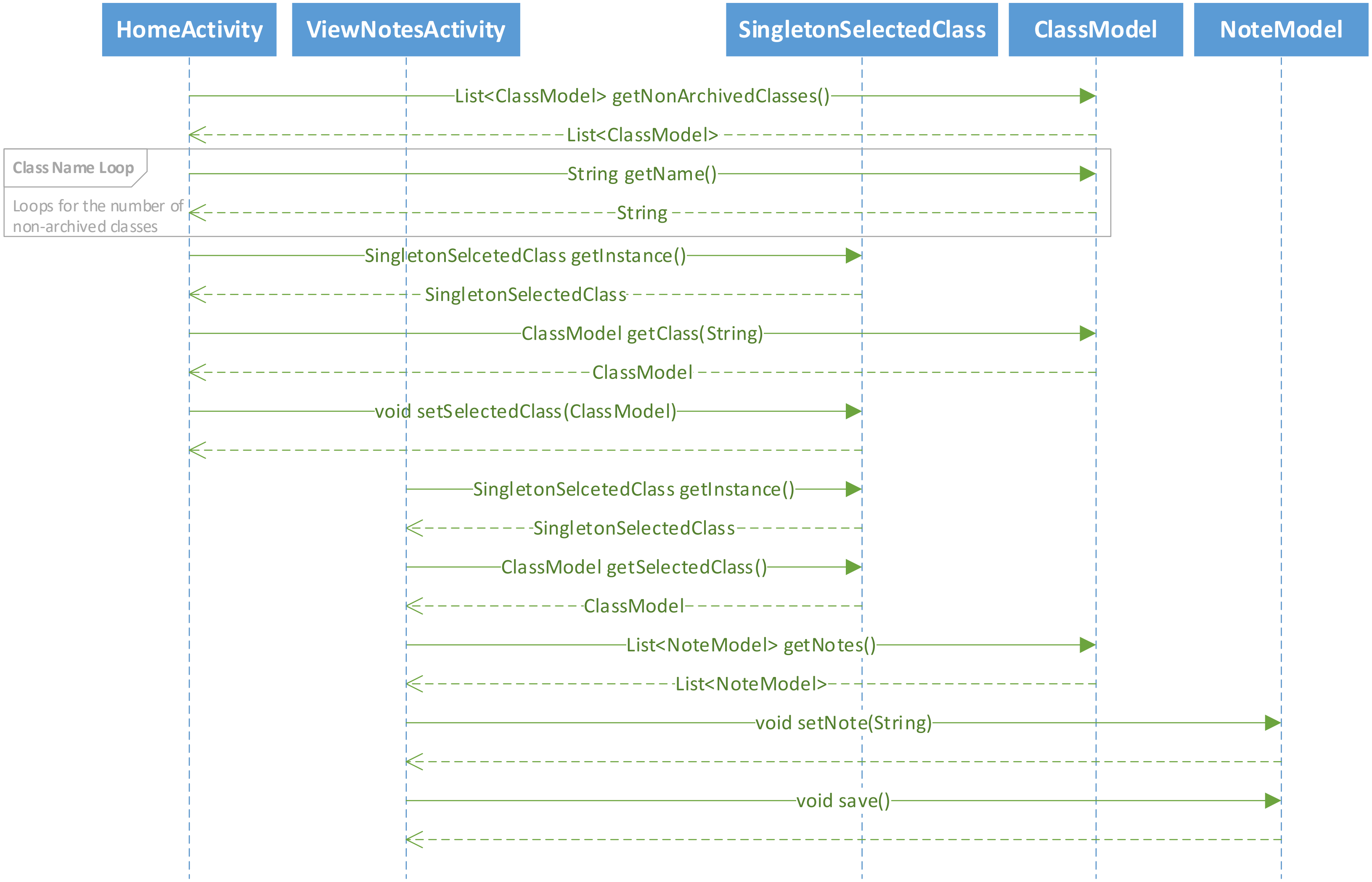
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
1	14	bytes			1 billion	Sqlite can't store more than 1 billion bytes per string.

UC 13 Edit Note Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that has the note he/she would like to edit.
    - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
    - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Notes button.
5. The system redirects the user to the Notes Activity.
6. The user taps the View All Notes button
7. The system redirects the user to the View Notes Activity.
    - The ViewNotesActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method. The ViewNotesActivity then uses the return value of the getSelectedClass() method to call the ClassModel's getNotes() method.
    - The List<NoteModel> that the getNotes() method returns represents all the notes that show up in the View Notes Activity.
8. The user taps on the note he/she would like to edit.
9. The system displays a pop up menu as described in requirement # 10.1
10. The user presses the View/Edit Note button
11. The system displays a large textbox with the contents of the note in it.
12. The user edits the note.
13. The user presses the Save Edits button.
    - The ViewNotesActivity calls the NoteModel's setNote(String) method using the NoteModel the user selected on step 8 and passes the user inputted note as a parameter to the method.
14. The system saves the edited note to the application.
    - The ViewNotesActivity calls the NoteModel's save() method (This is an ActiveAndroid method) updating the NoteModel's values in the database.

UC 13 Edit Note Sequence Diagram 01


HomeActivity
ViewNotesActivity
SingletonSelectedClass
ClassModel
NoteModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<NoteModel> getNotes()
List<NoteModel>
void setNote(String)
void save()

UC 14

General Information	
Use Case Name\Number: Listen to Audio 01 Subject Area: Listening to an audio recording. Description: The user wants to listen to an audio recording.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
7.1 & 7.1.1	Displays a pop up menu that allows the user to listen to an audio recording.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that he/she wants to listen to an audio recording from.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Audio Recordings button.		
5	The system redirects the user to the Audio Recordings Activity.		
6	The user taps the View Audio Recordings button.		
7	The system redirects the user to the View Audio Recordings Activity.		
8	The user taps the audio recording he/she would like to listen to		
9	The system displays a pop up menu as described in requirement # 7.1.		
10	The user presses the button to play the audio recording.		
11	The system starts playing the audio recording and displays a seek bar and a pause/play button for the user to use when listening to the audio.		

Exception Conditions

Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
N/A		

Post-Conditions

#	**Description**
1	The system starts playing the audio recording.
2	They system allows the user to pause/play the audio recording with the pause/play button.
3	The system allows the user to seek through the audio with the seek bar.

Candidate Objects

Class/Object Name	**Descriptions**	**Possible attributes**
ViewAudioRecordingsActivity	Handles the logic behind the UI of the View Audio Recordings Activity.	mSeekBar, mMediaPlayer

Assumptions

#	**Assumption**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
1	The user added a class to the app.	10/15/17	Matthew Del Fante		
2	The user added an audio recording to the app.	10/15/17	Matthew Del Fante		

Issues

#	**Issue**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
N/A					

Other Comments

Author	**Comment**	**Date**
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 1 (OR)Fixed:
Per: Hour: ☐ Day: ☐ Week: ☒ Month: ☐ Other:

Timing Information

#	**At/ Between**	**Step(s)**	**Timing Unit**	**Minimum**	**Average**	**Maximum**	**Comments**
N/A							

Volume Information

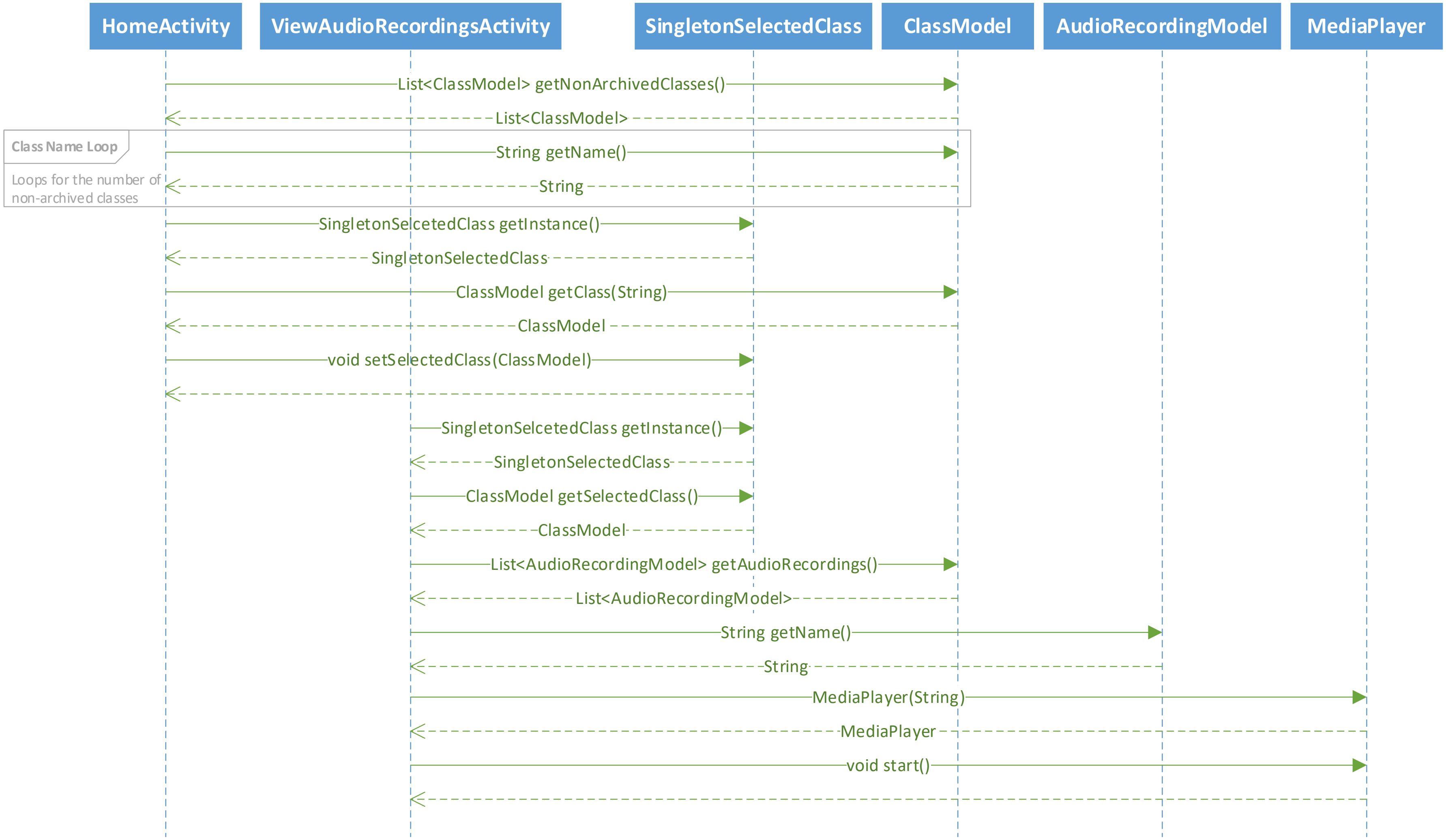
#	**Step #**	**Unit of Measure**	**Minimum**	**Average**	**Maximum**	**Comments**
N/A						

UC 14 Listen to Audio Recording Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that has the audio recording he/she would like to listen to.
    - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
    - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Audio Recordings button.
5. The system redirects the user to the Audio Recordings Activity.
6. The user taps the View Audio Recordings button.
    - The ViewAudioRecordingsActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method. The ViewAudioRecordingsActivity then uses the return value of the getSelectedClass() method to call the ClassModel's getAudioRecordings() method.
    - The List<AudioRecordingModel> that the getAudioRecordings () method returns represents all the audio recordings that show up in the View Audio Recordings Activity.
7. The system redirects the user to the View Audio Recordings Activity.
8. The user taps the audio recording he/she would like to listen to
9. The system displays a pop up menu as described in requirement # 7.1.
10. The user presses the button to play the audio recording.
    - The ViewAudioRecordingsActivity calls the AudioRecordingModel's getName() method using the AudioRecordingModel the user selected in step 8.
    - Now that the ViewAudioRecordingsActivity has the name of the audio recording, it constructs a MediaPlayer object passing the name of the audio recording as a parameter to the constructor of the MediaPlayer (Note: The MediaPlayer is an Android class).
11. The system starts playing the audio recording and displays a seek bar and a pause/play button for the user to use when listening to the audio.
    - The ViewAudioRecordingsActivity calls the MediaPlayer's start() method to start playing the audio recording.

UC 14 Listen to Audio Recording Sequence Diagram 01


HomeActivity
ViewAudioRecordingsActivity
SingletonSelectedClass
ClassModel
AudioRecordingModel
MediaPlayer
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<AudioRecordingModel> getAudioRecordings()
List<AudioRecordingModel>
String getName()
String
MediaPlayer(String)
MediaPlayer
void start()

UC 15

General Information

Use Case Name\Number: MAAC 01 Subject Area: Marking an assignment as completed. Description: The user wants to mark an assignment as completed after he/she completed the assignment.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace

REQ#	**Requirements Name and / or Short Description**
4.2	Navigates the user to the View Assignments Activity.
5.1.1.1	Allows a user to mark an assignment as completed.

Revision History

Author	**Date**	**Comments**
Matthew Del Fante	10/14/17	Initial draft.

Insertion Points in other Use Cases

Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors

Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions

#	Description
	See Use Case Summary.

Start Stimulus

The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps

Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that he/she wants to mark an assignment as completed from.		
3	The system redirects the user to the Class Activity.		
4	The user taps the View Assignments button.		
5	The system redirects the user to the View Assignments Activity.		
6	The user taps the assignment he/she wants to mark as completed.		
7	The system displays a pop up menu as described in requirement # 5.1.1.		
8	The user taps the Mark Assignment As Completed button.		
9	The system changes the background color of the assignment in the View Assignment Activity and Assignment Calendar to the color green.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
N/A		

Post-Conditions

#	Description
1	The system changes the background color of the assignment in the View Assignment Activity and Assignment Calendar to the color green.
2	The system makes the pop up menu disappear.
3	The system redirects the user to the View Assignments Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ViewAssignmentsActivity	Handles the logic behind the UI of the View Assignments Activity.	List<string> assignments

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/14/17	Matthew Del Fante		
2	The user added an assignment to the app.	10/14/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 4 (OR)Fixed:
Per: Hour: □ Day: □ Week: ☒ Month: □ Other:

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	at	9	millisecond			1	Changing the background color of the assignment in the Assignment Calendar and View Assignment Activity should be instantaneous.

Volume Information

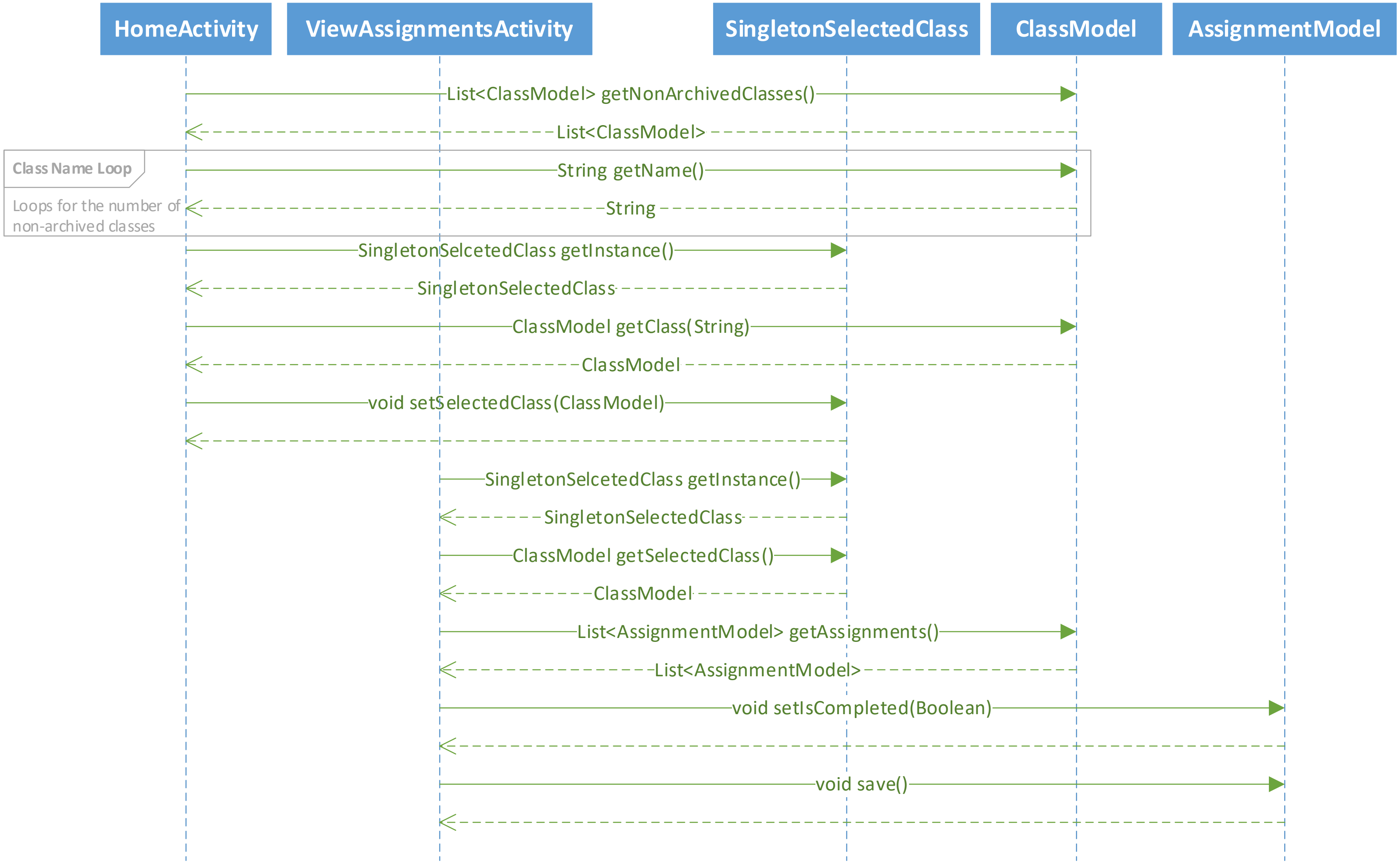
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 15 MAAC Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
   - The HomeActivity calls the ClassModel’s getNonArchivedClasses() method which returns a List<ClassModel>.
   - The HomeActivity then loops over the List<ClassModel> calling the ClassModel’s getName() method.
   - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes’ names.
2. The user taps the button of the class that he/she wants to mark an assignment as completed from.
   - The HomeActivity calls the SingletonSelectedClass’ getInstance() method.
   - The HomeActivity calls the ClassModel’s getClass(String) method passing the name of the class the user just tapped.
   - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass’ setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the View Assignments button.
5. The system redirects the user to the View Assignments Activity.
   - The ViewAssignmentsActivity calls the SingletonSelectedClass’ getInstance() method followed by the SingletonSelectedClass’ getSelectedClass() method. The ViewAssignmentsActivity then uses the return value of the getSelectedClass() method to call the ClassModel’s getAssignments() method.
   - The List<AssignmentModel> that the getAssignments() method returns represents all the assignments that show up in the View Assignments Activity.
6. The user taps the assignment he/she wants to mark as completed.
7. The system displays a pop up menu as described in requirement # 5.1.1.
8. The user taps the Mark Assignment As Completed button.
   - The ViewAssignmentsActivity calls the AssignmentModel’s setIsCompleted(Boolean) method using the AssignmentModel the user selected on step 6.
9. The system changes the background color of the assignment in the View Assignment Activity and Assignment Calendar to the color green.
   - The ViewAssignmentsActivity calls the AssignmentModel’s save() method (This is an ActiveAndroid method) updating the AssignmentModel’s values in the database.

UC 15 MAAC Sequence Diagram 01


HomeActivity
ViewAssignmentsActivity
SingletonSelectedClass
ClassModel
AssignmentModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<AssignmentModel> getAssignments()
List<AssignmentModel>
void setIsCompleted(Boolean)
void save()

UC 16

General Information	
Use Case Name\Number: Rename Audio Recording 01 Subject Area: Renaming an audio recording. Description: The user wants to rename an audio recording.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace

REQ#	Requirements Name and / or Short Description
6.2	Redirects the user to the View Audio Recordings Activity.
7.1 & 7.1.2	Allows the user to rename an audio recording.

Revision History

Author	Date	Comments
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases

Use Case Name	Use Case Number	Step Inserted After
N/A		

Actors

Actor Name	Person/System	Brief Description
See Use Case Summary.		

Pre-Conditions

#	Description
	See Use Case Summary.

Start Stimulus

The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps

Number	Description	Adds/Alt Name/Number	Bus Rule#
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that has the audio recording the user wants to rename.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Audio Recordings button.		
5	The system redirects the user to the Audio Recordings Activity.		
6	The user taps the View Audio Recordings button.		
7	The system redirects the user to the View Audio Recordings Activity.		
8	The user taps the audio recording he/she would like to rename.		
9	The system displays a pop up menu as described in requirement # 7.1.		
10	The user presses the Rename Audio Recording button.		
11	The system displays a textbox along with done and cancel buttons to rename the audio recording.		
12	The user renames the audio recording and presses the done button.		
13	The system renames the audio recording.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user presses the cancel button when renaming the audio recording.	The audio recording won't be renamed and the system will display the pop up menu in step 9.	
The user tries to rename the audio recording with only whitespace.	The done button will be greyed out and unclickable.	
The user enters a non-unique audio recording name.	The system will notify the user that he/she already has an audio recording with that name and prompt the user to enter in a unique name.	

Post-Conditions

#	Description
1	The system replaces all instances of the old audio recording name with the new name.
2	The system makes the textbox and the pop up menu disappear.
3	The system redirects the user to the View Audio Recordings Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ViewAudioRecordingsActivity	Handles the logic behind the UI of the View Audio Recordings Activity.	List<string> audioNames

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/15/17	Matthew Del Fante		
2	The user added an audio recording to the app.	10/15/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency:	Minimum: 0	Maximum:	Average: 4	(OR)Fixed:	
Per:	Hour: ☐	Day: ☐	Week: ☐	Month: ☒	Other:

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	at	13	milliseconds			1	Renaming the audio recording should be instantaneous with respect to the UI.

Volume Information

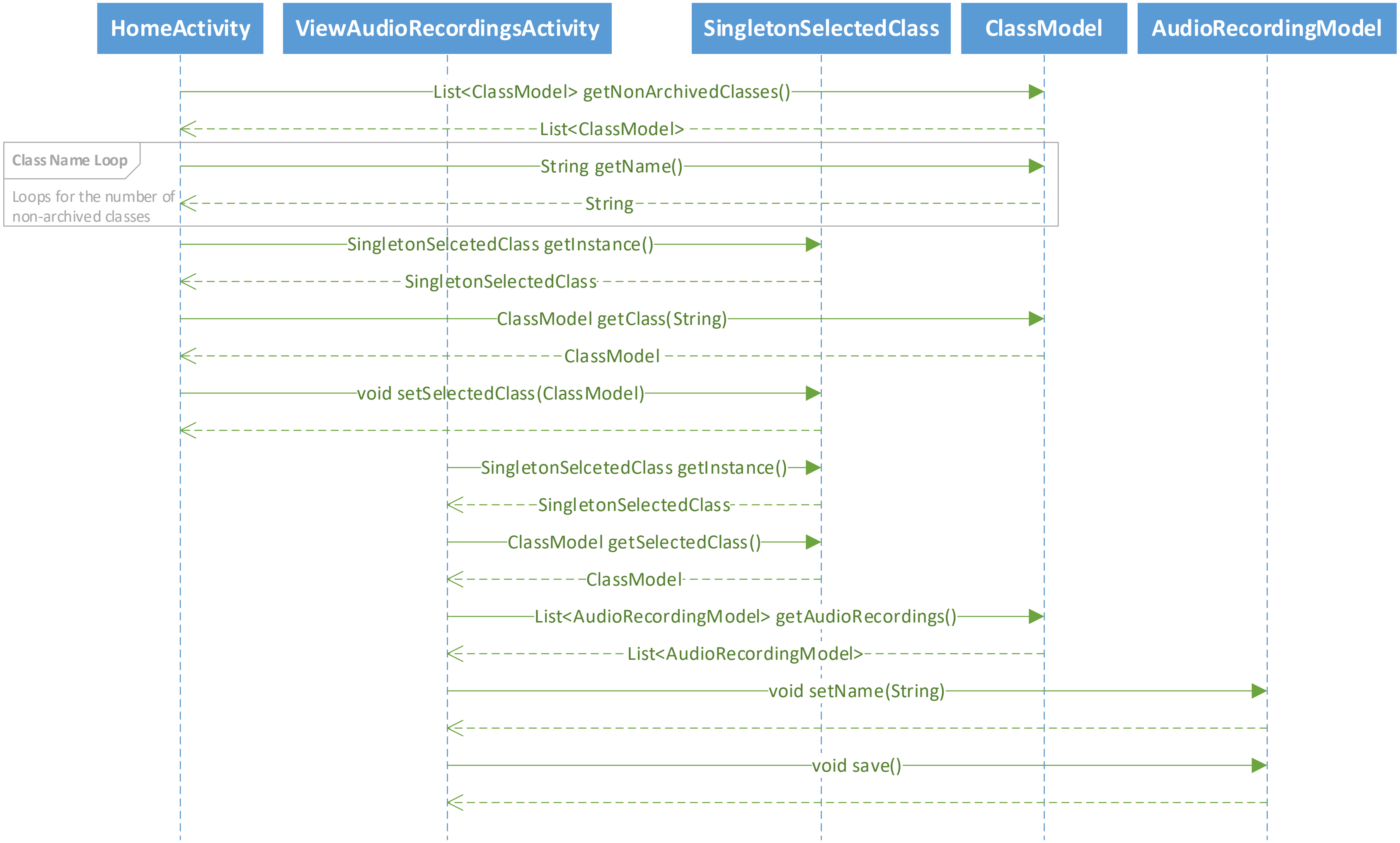
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 16 Rename Audio Recording Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that has the audio recording the user wants to rename
    - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
    - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Audio Recordings button.
5. The system redirects the user to the Audio Recordings Activity.
6. The user taps the View Audio Recordings button.
    - The ViewAudioRecordingsActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method. The ViewAudioRecordingsActivity then uses the return value of the getSelectedClass() method to call the ClassModel's getAudioRecordings() method.
    - The List<AudioRecordingModel> that the getAudioRecordings () method returns represents all the audio recordings that show up in the View Audio Recordings Activity.
7. The system redirects the user to the View Audio Recordings Activity.
8. The user taps the audio recording he/she would like to rename.
9. The system displays a pop up menu as described in requirement # 7.1.
10. The user presses the Rename Audio Recording button.
11. The system displays a textbox along with done and cancel buttons to rename the audio recording.
12. The user renames the audio recording and presses the done button.
    - The ViewAudioRecordingsActivity calls the AudioRecordingModel's setName(String) method using the AudioRecordingModel the user selected in step 8 and passing the user inputted name as a parameter to the method.
13. The system renames the audio recording.
    - The AudioRecordingsActivity calls the AudioRecordingModel's save() method (This is an ActiveAndroid method) updating the AudioRecordingModel's name in the database.

UC 16 Rename Audio Recording Sequence Diagram 01


HomeActivity
ViewAudioRecordingsActivity
SingletonSelectedClass
ClassModel
AudioRecordingModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<AudioRecordingModel> getAudioRecordings()
List<AudioRecordingModel>
void setName(String)
void save()

UC 17

General Information	
Use Case Name\Number: Rename Class 01 Subject Area: Renaming a class. Description: The user wants to rename a class he/she added to the Class Hub app.	Responsible Analyst : Matthew Del Fante

Requirements/Feature Trace

REQ#	Requirements Name and / or Short Description
3.4.1.2 & 3.4.1.2.1	The user long presses a class to display a pop up menu
3.4.1.2.1.1	The user renamed the class and the new class name is displayed within the app
3.4.1.2.1.4.	The done button is only allowed to be pressed when a character is typed into Edit Class Name text box.

Revision History

Author	Date	Comments
Matthew Del Fante	10/13/17	Initial Draft.

Insertion Points in other Use Cases

Use Case Name	Use Case Number	Step Inserted After
N/A		

Actors

Actor Name	Person/System	Brief Description
See Use Case Summary.		

Pre-Conditions

#	Description
	See Use Case Summary.

Start Stimulus

The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps

Number	Description	Adds/Alt Name/Number	Bus Rule#
1	The system takes the user to the Class Hub Home Activity.		
2	The user long presses on the name of a class.		
3	The system displays a pop up menu as described in requirement # 3.4.1.2.1.		
4	The user types a new class name into the Edit Class Name textbox.		
5	The user presses the done button.		
6	The system makes the pop up menu disappear.		
7	The system replaces all instances of the old class name with the new class name.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user tries to press the done button before typing any letters in the text box.	The done button is greyed out and unclickable.	
The user enters a non-unique class name.	The system will notify the user that he/she already has a class with that name.	

Post-Conditions

#	Description
1	The system replaces all instances of the old class name with the new class name.
2	The system makes the pop up menu disappear.
3	The system displays the home Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
HomeActivity	Handles the logic behind the UI of the Home Activity.	List<string> classNames

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/13/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency:	Minimum: 1		Maximum:	Average: 4	(OR)Fixed:
Per:	Hour: □	Day: □	Week: □	Month: □	Other: Year

Timing Information

#	At / Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	After	7	milliseconds			1	Replacing the old class name on the Home Activity with the new class name should be instantaneous.

Volume Information

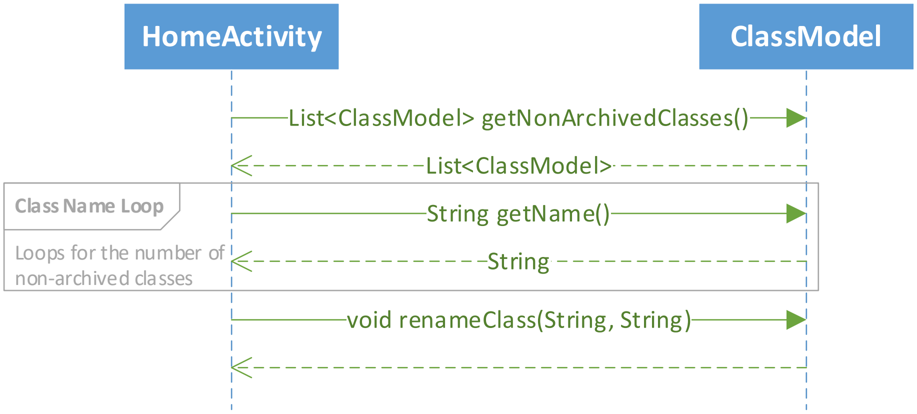
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 17 Rename Class Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user long presses on the name of a class.
3. The system displays a pop up menu as described in requirement # 3.4.1.2.1.
4. The user types a new class name into the Edit Class Name textbox.
5. The user presses the done button.
6. The system makes the pop up menu disappear.
7. The system replaces all instances of the old class name with the new class name.
    - The HomeActivity calls the ClassModel's renameClass(String, String) method passing the name of the class the user long pressed in step 2 as the first parameter and passing the name the user typed in step 4 as the second parameter to the method. The ClassModel's name is then updated in the database.

UC 17 Rename Class Sequence Diagram 01


HomeActivity
ClassModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
void renameClass(String, String)

UC 18

General Information	
Use Case Name\Number: Rename Note 01 Subject Area: Renaming a note. Description: The user wants to rename a note he/she saved.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
9.2	Redirects the user to the View Notes Activity
10.1 & 10.1.2	Displays a pop up menu that allows a user to rename a note.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that has the note he/she would like to rename.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Notes button.		
5	The system redirects the user to the Notes Activity.		
6	The user taps the View All Notes button		
7	The system redirects the user to the View Notes Activity.		
8	The user taps on the note he/she would like to rename.		
9	The system displays a pop up menu as described in requirement # 10.1.		
10	The user presses the Rename Note button		
11	The system displays a textbox, a done button and a cancel button.		
12	The user types in a new note name into the textbox and presses the done button.		
13	The system renames the note in the application.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
The user presses the cancel button when renaming the note.	The note won't be renamed and the system will display the pop up menu in step 9.	
The user tries to rename the note with only whitespace.	The done button will be greyed out and unclickable.	
The user enters a non-unique note name.	The system will notify the user that he/she already has a note with that name and prompt the user to enter in a unique name.	

Post-Conditions

#	Description
1	The system replaces all instances of the old note name with the new name.
2	The system makes the textbox and the pop up menu disappear.
3	The system redirects the user to the View Notes Activity.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ViewNotesActivity	Handles the logic behind the UI of the View Notes Activity.	List<string> mNoteNames

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/15	Matthew Del Fante		
2	The user added a note to a class.	10/15	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 4 (OR)Fixed:

Per: Hour: □ Day: □ Week: □ Month: □ Other: School Term

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	at	13	milliseconds			1	Renaming note in the UI should be instantaneous.

Volume Information

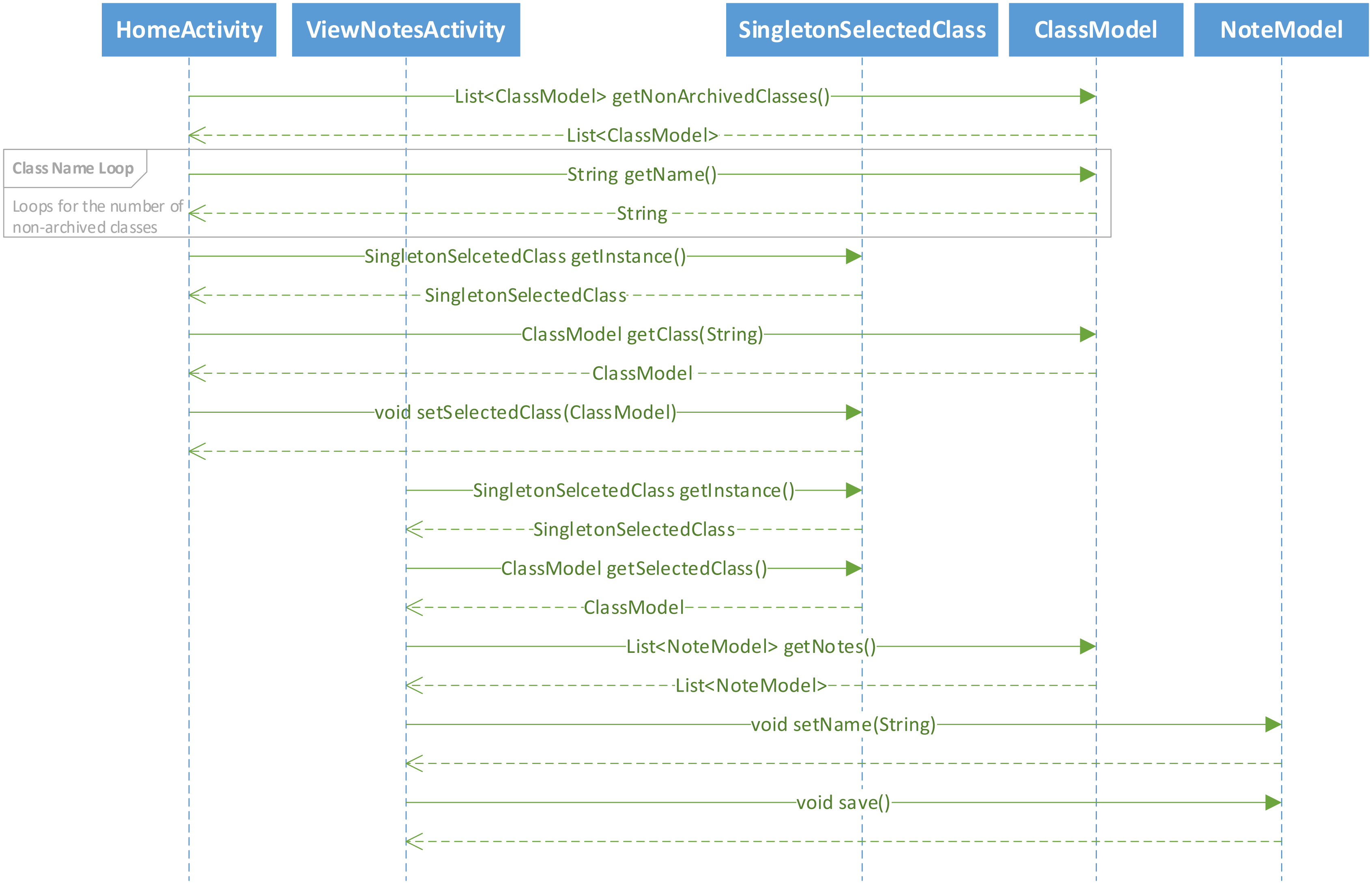
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 18 Rename Note Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
   - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
   - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
   - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that has the note he/she would like to rename.
   - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
   - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
   - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Notes button.
5. The system redirects the user to the Notes Activity.
6. The user taps the View All Notes button
7. The system redirects the user to the View Notes Activity.
   - The ViewNotesActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method. The ViewNotesActivity then uses the return value of the getSelectedClass() method to call the ClassModel's getNotes() method.
   - The List<NoteModel> that the getNotes() method returns represents all the notes that show up in the View Notes Activity.
8. The user taps on the note he/she would like to rename.
9. The system displays a pop up menu as described in requirement # 10.1.
10. The user presses the Rename Note button
11. The system displays a textbox, a done button and a cancel button.
12. The user types in a new note name into the textbox and presses the done button.
    - The ViewNotesActivity calls the NoteModel's setName(String) method using the NoteModel the user selected on step 8 and passes the user inputted name a parameter to the method.
13. The system renames the note in the application.
    - The ViewNotesActivity calls the NoteModel's save() method (This is an ActiveAndroid method) updating the NoteModel's values in the database.

UC 18 Rename Note Sequence Diagram 01


HomeActivity
ViewNotesActivity
SingletonSelectedClass
ClassModel
NoteModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<NoteModel> getNotes()
List<NoteModel>
void setName(String)
void save()

UC 19

General Information

Use Case Name\Number: Rename Video Recording 01 Subject Area: Renaming a video recording. Description: The user wants to rename a video recording.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace

REQ#	Requirements Name and / or Short Description
8	Encompasses everything that has to do with videos in Class Hub.

Revision History

Author	Date	Comments
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases

Use Case Name	Use Case Number	Step Inserted After
N/A		

Actors

Actor Name	Person/System	Brief Description
See Use Case Summary.		

Pre-Conditions

#	Description
	See Use Case Summary.

Start Stimulus

The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps

Number	Description	Adds/Alt Name/Number	Bus Rule#
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that has the video recording he/she would like to rename.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Video Recordings button.		
5	The system redirects the user to the Video Recordings Activity.		
6	The user taps the View Video Recordings button.		
7	The system redirects the user to the View Video Recordings Activity.		
8	The user taps the video recording he/she would like to rename.		
9	The system displays a pop up menu as described in requirement # 7.1, but with video information instead of audio information.		
10	The user presses the Rename Video Recording button.		
11	The system displays a textbox, a done button and a cancel button.		
12	The user types a new name in the textbox and presses the done button.		

13	The system renames the video recording in the application.		

Exception Conditions

Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
The user presses the cancel button when renaming the video recording.	The video recording won't be renamed and the system will display the pop up menu in step 9.	
The user tries to rename the video recording with only whitespace.	The done button will be greyed out and unclickable.	
The user enters a non-unique video recording name.	The system will notify the user that he/she already has a video recording with that name and prompt the user to enter in a unique name.	

Post-Conditions

#	**Description**
1	The system replaces all instances of the old video recording name with the new name.
2	The system makes the textbox and the pop up menu disappear.
3	The system redirects the user to the View Video Recordings Activity.

Candidate Objects

Class/Object Name	**Descriptions**	**Possible attributes**
ViewVideoRecordingsActivity	Handles the logic behind the UI of the View Video Recordings Activity.	List<string> videoNames

Assumptions

#	**Assumption**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
1	The user added a class to the app.	10/15/17	Matthew Del Fante		
2	The user added a video recording to the app.	10/15/17	Matthew Del Fante		

Issues

#	**Issue**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
N/A					

Other Comments

Author	**Comment**	**Date**
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 4 (OR)Fixed:
Per: Hour: □ Day: □ Week: □ Month: ☒ Other:

Timing Information

#	**At/ Between**	**Step(s)**	**Timing Unit**	**Minimum**	**Average**	**Maximum**	**Comments**
1	at	13	milliseconds			1	Renaming the video recording in the UI should be instantaneous.

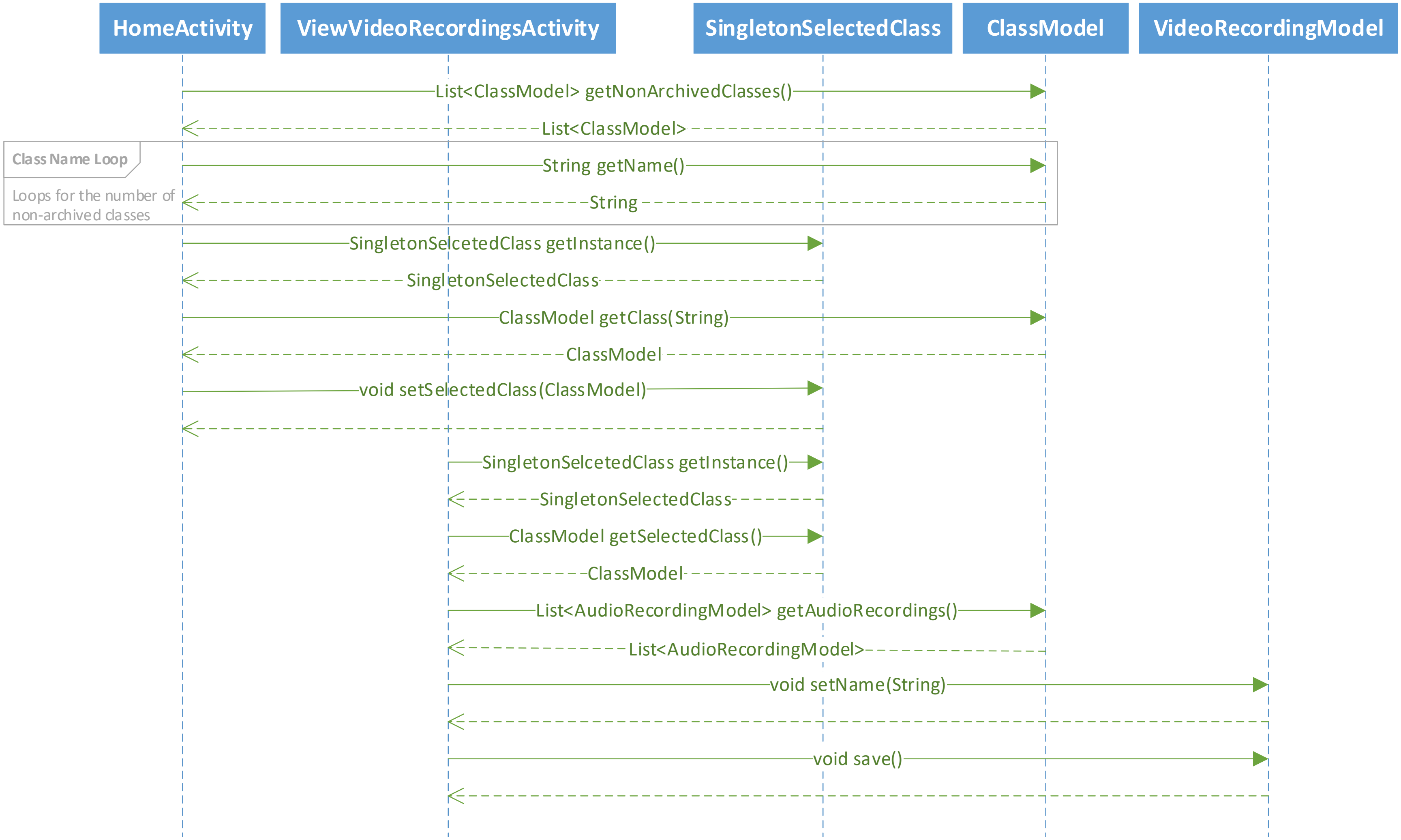
Volume Information						
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 19 Rename Video Recording Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
   - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
   - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
   - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that has the video recording he/she would like to rename.
   - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
   - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
   - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Video Recordings button.
5. The system redirects the user to the Video Recordings Activity.
6. The user taps the View Video Recordings button.
7. The system redirects the user to the View Video Recordings Activity.
   - The ViewVideoRecordingsActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method. The ViewVideoRecordingsActivity then uses the return value of the getSelectedClass() method to call the ClassModel's getVideoRecordings() method.
   - The List<VideoRecordingModel> that the getVideoRecordings() method returns represents all the video recordings that show up in the View Video Recordings Activity.
8. The user taps the video recording he/she would like to rename.
9. The system displays a pop up menu as described in requirement # 7.1, but with video information instead of audio information.
10. The user presses the Rename Video Recording button.
11. The system displays a textbox, a done button and a cancel button.
12. The user types a new name in the textbox and presses the done button.
    - The ViewVideoRecordingsActivity calls the VideoRecordingModel's setName(String) method using the VideoRecordingModel the user selected on step 8 and passes the user inputted name a parameter to the method.
13. The system renames the video recording in the application.
    - The ViewVideoRecordingsActivity calls the VideoRecordingModel's save() method (this is an ActiveAndroid method) updating the VideoRecordingModel's values in the database.

UC 19 Rename Video Recording Sequence Diagram 01


HomeActivity
ViewVideoRecordingsActivity
SingletonSelectedClass
ClassModel
VideoRecordingModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<AudioRecordingModel> getAudioRecordings()
List<AudioRecordingModel>
void setName(String)
void save()

UC 20

General Information	
Use Case Name\Number: Retrieve Archived Classes 01 Subject Area: Retrieving all archived classes. Description: The user wants to retrieve all archived classes and add them to the application.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
3.3.1.2	All of the logic for retrieving archived classes and inserting them into the application.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/13/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user long presses the Add Class button.		
3	The system displays a pop up menu according to requirement # 3.3.1.2.1.		
4	The user taps the Retrieve Archived Classes button.		
5	The system pulls all of the archived classes' information from the database and adds them to the application.		

Exception Conditions		
Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
The user has not archived any classes yet.	The pop up menu disappears and nothing happens.	
The user taps the cancel button.	The pop up menu disappears and nothing happens.	

Post-Conditions

#	Description
1	The system pulls all of the archived classes' information from the database and adds it to the application.
2	The system makes the pop up menu disappear.
3	The system redirects the user to the Home Activity, which now will be populated with previously archived classes information.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
HomeActivity	Handles the logic behind the UI of the Home Activity.	List<string> classNames

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user has at least one class archived.	10/13/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 1 (OR)Fixed:

Per: Hour: ☐ Day: ☐ Week: ☐ Month: ☐ Other: Year

Timing Information

#	At / Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
1	at	5	milliseconds			5	Updating the UI to include archived class information should almost be instantons.

Volume Information

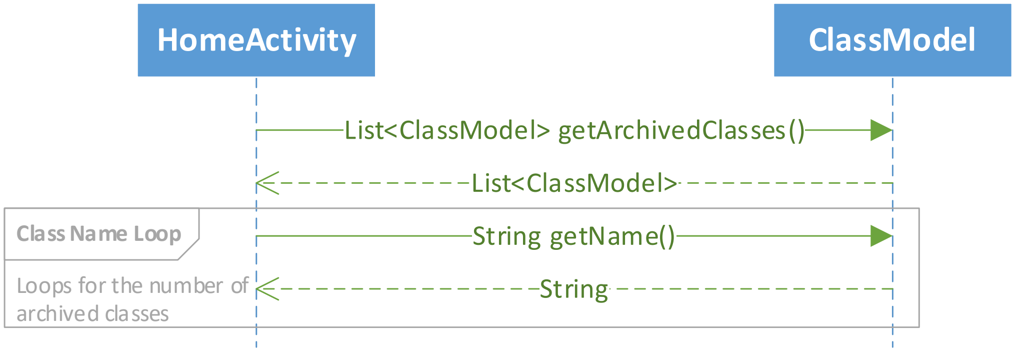
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
1	5	bytes			1 billion	The max size of a Sqlite string is 1 billion bytes. Thus, the system shall not retrieve more than 1 billion bytes of data.

UC 20 Retrieve Archived Classes Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
2. The user long presses the Add Class button.
3. The system displays a pop up menu according to requirement # 3.3.1.2.1.
4. The user taps the Retrieve Archived Classes button.
5. The system pulls all of the archived classes' information from the database and adds them to the application.
    - The HomeActivity calls the ClassModel's getArchivedClasses() method which returns a List<ClassModel> of all the archived classes.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons with the archived classes names.

UC 20 Retrieve Archived Classes Sequence Diagram 01


HomeActivity
ClassModel
List<ClassModel> getArchivedClasses()
List<ClassModel>
Class Name Loop
String getName()
Loops for the number of archived classes
String

UC 21

General Information	
Use Case Name\Number: View Assignment 01 Subject Area: Viewing an assignment's information. Description: The user wants to view an assignment's information.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
4.2	Navigates the user to the View Assignments Activity.
5.1	Allows a user to view an assignment's information.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/14/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that he/she wants to view an assignment from.		
3	The system redirects the user to the Class Activity.		
4	The user taps the View Assignments button.		
5	The system redirects the user to the View Assignments Activity.		
6	The user taps the assignment he/she wants to view the assignment information from.		
7	The system displays a pop up menu as described in requirement # 5.1.1.		
8	The user taps on any of the buttons or text fields to see the assignment's information.		

Exception Conditions		
Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
N/A		

Post-Conditions

#	Description
1	The user can now see an assignment's information (name, due date, priority level, notes).

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ViewAssignmentsActivity	Handles the logic behind the UI of the View Assignments Activity.	List<string> assignments

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/14/17	Matthew Del Fante		
2	The user added an assignment to the app.	10/14/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 4 (OR)Fixed:
Per: Hour: □ Day: □ Week: ☒ Month: □ Other:

Timing Information

#	At / Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
N/A							

Volume Information

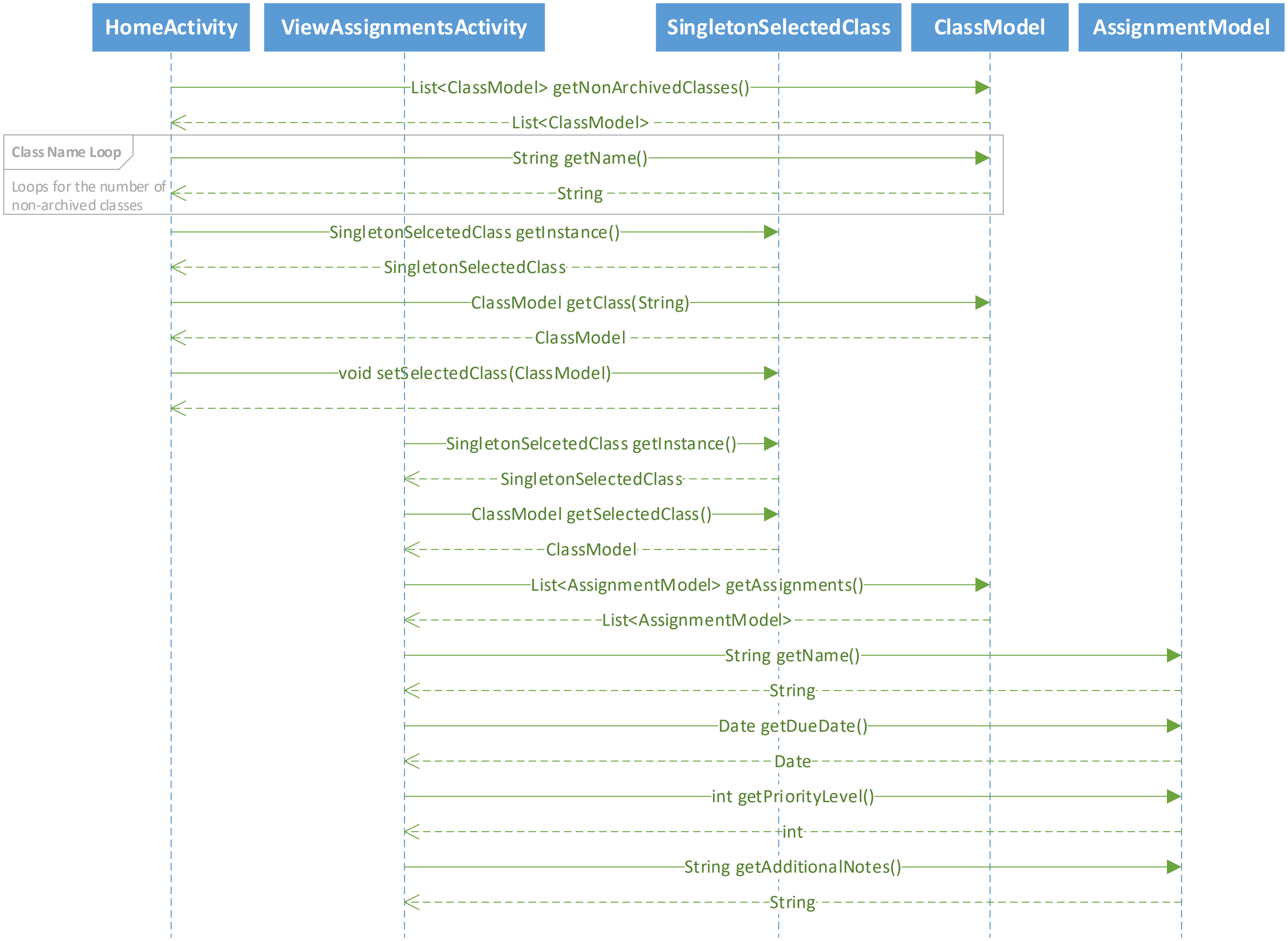
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 21 View Assignment Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
   - The HomeActivity calls the ClassModel’s getNonArchivedClasses() method which returns a List<ClassModel>.
   - The HomeActivity then loops over the List<ClassModel> calling the ClassModel’s getName() method.
   - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes’ names.
2. The user taps the button of the class that he/she wants to view an assignment from.
   - The HomeActivity calls the SingletonSelectedClass’ getInstance() method.
   - The HomeActivity calls the ClassModel’s getClass(String) method passing the name of the class the user just tapped.
   - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass’ setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the View Assignments button.
5. The system redirects the user to the View Assignments Activity.
   - The ViewAssignmentsActivity calls the SingletonSelectedClass’ getInstance() method followed by the SingletonSelectedClass’ getSelectedClass() method. The ViewAssignmentsActivity then uses the return value of the getSelectedClass() method to call the ClassModel’s getAssignments() method.
   - The List<AssignmentModel> that the getAssignments() method returns represents all the assignments that show up in the View Assignments Activity.
6. The user taps the assignment he/she wants to view the assignment information from.
7. The system displays a pop up menu as described in requirement # 5.1.1.
8. The user taps on any of the buttons or text fields to see the assignment’s information.
   - The ViewAssignmentsActivity uses the AssignmentModel the user tapped in step 8 to call the AssignmentModel’s getName(), getDueDate(), getPriorityLevel() and getAdditionalNotes() methods to show the assignment’s information.

UC 21 View Assignment Sequence Diagram 01


HomeActivity
ViewAssignmentsActivity
SingletonSelectedClass
ClassModel
AssignmentModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<AssignmentModel> getAssignments()
List<AssignmentModel>
String getName()
String
Date getDueDate()
Date
int getPriorityLevel()
int
String getAdditionalNotes()
String

UC 22

General Information	
Use Case Name\Number: View Notes 01 Subject Area: Viewing Notes. Description: The user wants to view a note he/she wrote about a class.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
9.2	Redirects the user to the View Notes Activity.
10.1.1	Allows a user to view a note.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that has the note he/she would like to view.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Notes button.		
5	The system redirects the user to the Notes Activity.		
6	The user taps the View All Notes button		
7	The system redirects the user to the View Notes Activity.		
8	The user taps on the note he/she would like to view.		
9	The system displays a pop up menu as described in requirement # 10.1.		
10	The user presses the View/Edit Note button		
11	The system displays a large textbox with the contents of the note in it.		

Exception Conditions

Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
N/A		

Post-Conditions

#	**Description**
1	The user can view the note.

Candidate Objects

Class/Object Name	**Descriptions**	**Possible attributes**
ViewNotesActivity	Handles the logic behind the UI of the View Notes Activity.	List<string> mNotes

Assumptions

#	**Assumption**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
1	The user added a class to the app.	10/15	Matthew Del Fante		
2	The user added a note to a class.	10/15	Matthew Del Fante		

Issues

#	**Issue**	**Date Raised**	**Raised By**	**Date Verified**	**Verified By**
N/A					

Other Comments

Author	**Comment**	**Date**
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 4 (OR)Fixed:

Per: Hour: □ Day: □ Week: □ Month: □ Other: School term.

Timing Information

#	**At/ Between**	**Step(s)**	**Timing Unit**	**Minimum**	**Average**	**Maximum**	**Comments**
N/A							

Volume Information

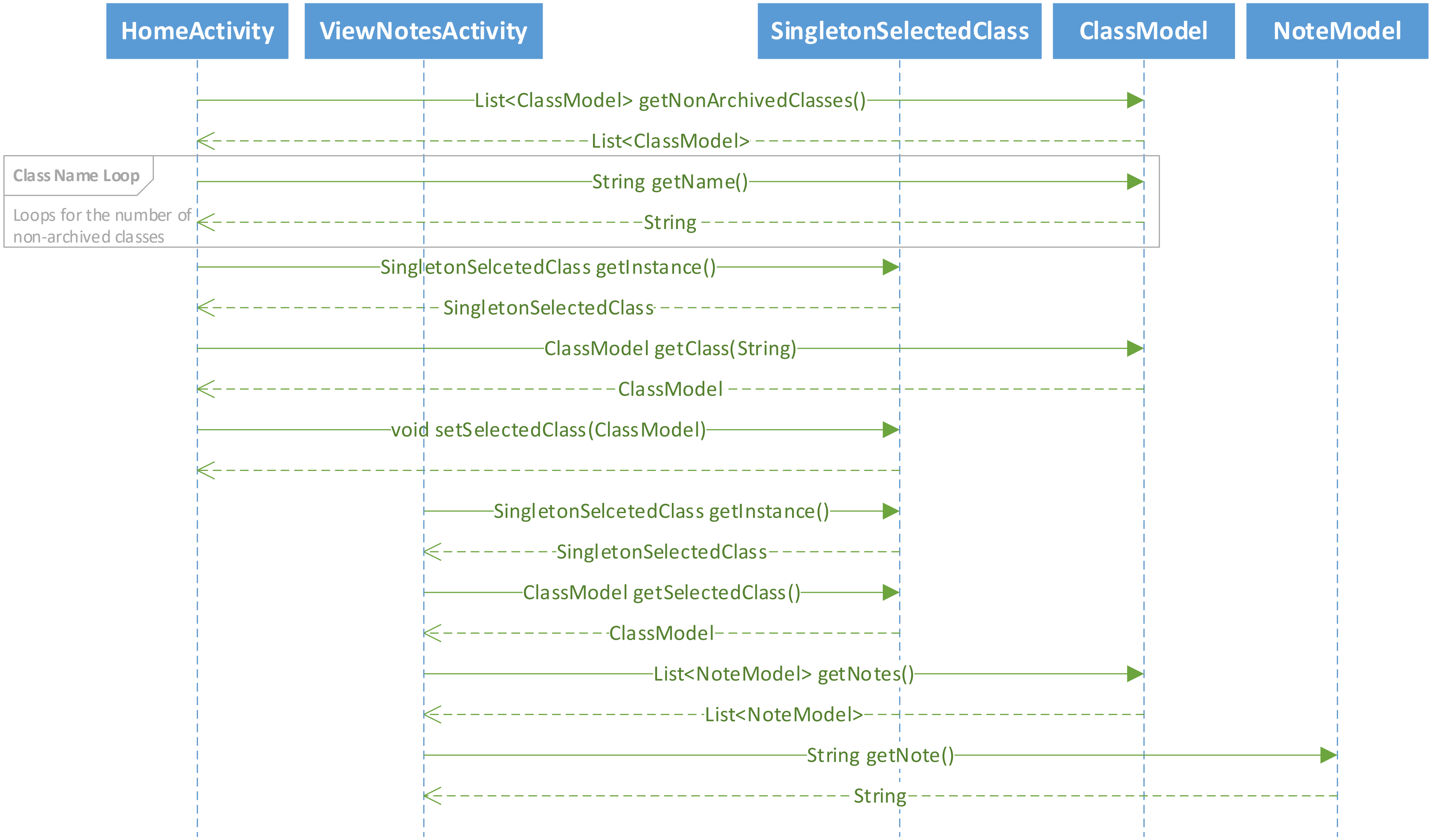
#	**Step #**	**Unit of Measure**	**Minimum**	**Average**	**Maximum**	**Comments**
N/A						

UC 22 View Notes Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
    - The HomeActivity calls the ClassModel’s getNonArchivedClasses() method which returns a List<ClassModel>.
    - The HomeActivity then loops over the List<ClassModel> calling the ClassModel’s getName() method.
    - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes’ names.
2. The user taps the button of the class that has the note he/she would like to view.
    - The HomeActivity calls the SingletonSelectedClass’ getInstance() method.
    - The HomeActivity calls the ClassModel’s getClass(String) method passing the name of the class the user just tapped.
    - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass’ setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Notes button.
5. The system redirects the user to the Notes Activity.
6. The user taps the View All Notes button
7. The system redirects the user to the View Notes Activity.
    - The ViewNotesActivity calls the SingletonSelectedClass’ getInstance() method followed by the SingletonSelectedClass’ getSelectedClass() method. The ViewNotesActivity then uses the return value of the getSelectedClass() method to call the ClassModel’s getNotes() method.
    - The List<NoteModel> that the getNotes() method returns represents all the notes that show up in the View Notes Activity.
8. The user taps on the note he/she would like to view.
9. The system displays a pop up menu as described in requirement # 10.1.
10. The user presses the View/Edit Note button
11. The system displays a large textbox with the contents of the note in it.
    - The ViewNotesActivity calls the NoteModel’s getNote() method using the NoteModel the user tapped in step 8 to display the note.

UC 22 View Notes Sequence Diagram 01


HomeActivity
ViewNotesActivity
SingletonSelectedClass
ClassModel
NoteModel
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<NoteModel> getNotes()
List<NoteModel>
String getNote()
String

UC 23

General Information	
Use Case Name\Number: Watch Video Recording 01 Subject Area: Watching a video recording. Description: The user wants to watch a video recording.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
8	Encompasses everything that has to do with videos in Class Hub.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/15/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The user taps the button of the class that has the video the user would like to watch.		
3	The system redirects the user to the Class Activity.		
4	The user taps the Video Recordings button.		
5	The system redirects the user to the Video Recordings Activity.		
6	The user taps the View Video Recordings button.		
7	The system redirects the user to the View Video Recordings Activity.		
8	The user taps the video recording he/she would like to watch.		
9	The system displays a pop up menu as described in requirement # 7.1, but with video information instead of audio information.		
10	The user presses the button to play the video recording.		
11	The system starts playing the video recording and displays a seek bar along with a pause/play button for the user to use when watching the video.		

Exception Conditions

Exception Situations	Action(s) on Exception	Adds\Alt UC #
N/A		

Post-Conditions

#	Description
1	The system starts playing the video recording.
2	They system allows the user to pause/play the video recording with the pause/play button.
3	The system allows the user to seek through the video with the seek bar.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
ViewVideoRecordingsActivity	Handles the logic behind the UI of the View Video Recordings Activity.	mSeekBar, mMediaPlayer, List<string> videoNames

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user added a class to the app.	10/15/17	Matthew Del Fante		
2	The user added a video recording to the app.	10/15/17	Matthew Del Fante		

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency: Minimum:0 Maximum: Average: 1 (OR)Fixed:

Per: Hour: □ Day: □ Week: □ Month: ☒ Other:

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
N/A							

Volume Information

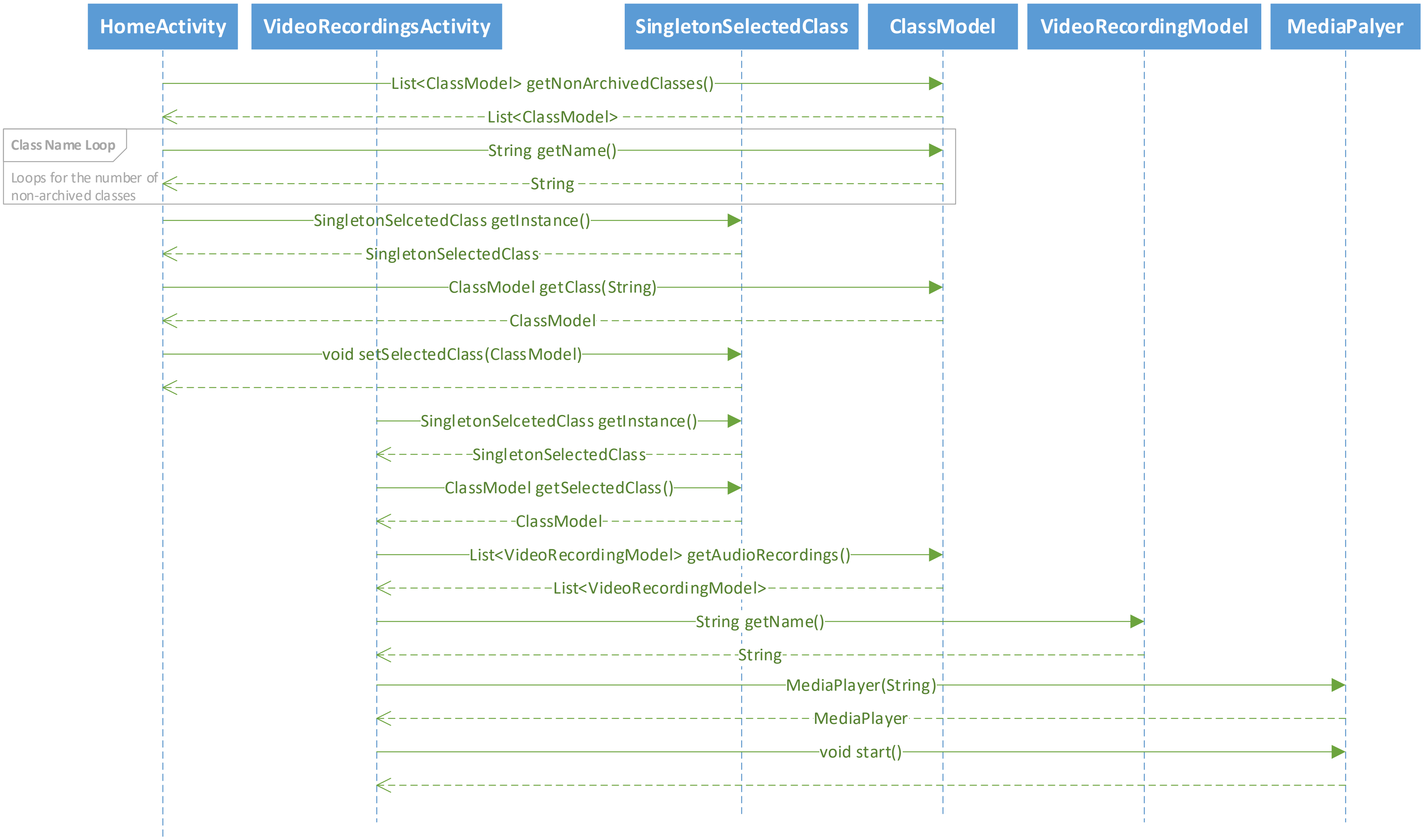
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 23 Watch Video Recording Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
   - The HomeActivity calls the ClassModel's getNonArchivedClasses() method which returns a List<ClassModel>.
   - The HomeActivity then loops over the List<ClassModel> calling the ClassModel's getName() method.
   - Using the return value of the getName() method, the HomeActivity creates buttons that have the non-archived classes' names.
2. The user taps the button of the class that has the video the user would like to watch.
   - The HomeActivity calls the SingletonSelectedClass' getInstance() method.
   - The HomeActivity calls the ClassModel's getClass(String) method passing the name of the class the user just tapped.
   - The HomeActivity passes the return value of the of the getClass(String) method as a parameter to the SingletonSelectedClass' setSelectedClass(ClassModel) method.
3. The system redirects the user to the Class Activity.
4. The user taps the Video Recordings button.
5. The system redirects the user to the Video Recordings Activity.
6. The user taps the View Video Recordings button.
7. The system redirects the user to the View Video Recordings Activity.
   - The ViewVideoRecordingsActivity calls the SingletonSelectedClass' getInstance() method followed by the SingletonSelectedClass' getSelectedClass() method. The ViewVideoRecordingsActivity then uses the return value of the getSelectedClass() method to call the ClassModel's getVideoRecordings() method.
   - The List<VideoRecordingModel> that the getVideoRecordings() method returns represents all the video recordings that show up in the View Video Recordings Activity.
8. The user taps the video recording he/she would like to watch.
9. The system displays a pop up menu as described in requirement # 7.1, but with video information instead of audio information.
10. The user presses the button to play the video recording.
    - The ViewVideoRecordingsActivity calls the VideoRecordingModel 's getName() method using the VideoRecordingModel the user selected in step 8.
    - Now that the ViewVideoRecordingsActivity has the name of the video recording, it constructs a MediaPlayer object passing the name of the video recording as a parameter to the constructor of the MediaPlayer (Note: The MediaPlayer is an Android class).
11. The system starts playing the video recording and displays a seek bar along with a pause/play button for the user to use when watching the video.
    - The ViewVideoRecordingsActivity calls the MediaPlayer's start() method to start playing the video recording.

UC 23 Watch Video Recording Sequence Diagram 01


HomeActivity
VideoRecordingsActivity
SingletonSelectedClass
ClassModel
VideoRecordingModel
MediaPalyer
List<ClassModel> getNonArchivedClasses()
List<ClassModel>
Class Name Loop
Loops for the number of non-archived classes
String getName()
String
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getClass(String)
ClassModel
void setSelectedClass(ClassModel)
SingletonSelcetedClass getInstance()
SingletonSelectedClass
ClassModel getSelectedClass()
ClassModel
List<VideoRecordingModel> getAudioRecordings()
List<VideoRecordingModel>
String getName()
String
MediaPlayer(String)
MediaPlayer
void start()

UC 24

General Information	
Use Case Name\Number: VAIAC 01 Subject Area: Viewing assignments in the assignment calendar. Description: The user would like to view assignments in the assignment calendar.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace

REQ#	**Requirements Name and / or Short Description**
3.2.1.1	Explains how many days the Assignment Calendar shows.
3.2.1.2	Explains how the Assignment Calendar is scrollable.
3.2.1.3	Explains how assignments show up in the Assignment Calendar.

Revision History

Author	**Date**	**Comments**
Matthew Del Fante	10/16/17	Initial draft.

Insertion Points in other Use Cases

Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors

Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions

#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps

Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The system displays the Assignment Calendar on the Home Activity.		
3	The Assignment Calendar functions as described in requirement # 3.2.1.		
4	The user can see the assignments in the assignment calendar.		

Exception Conditions

Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
The user hasn't added an assignment to a class.	The user won't see any assignments in the Assignment Calendar.	

Post-Conditions

#	Description
1	The user can see the assignments in the assignment calendar.
2	The user can scroll through the assignment calendar to see assignments that are due in the future and assignments that were due in the past.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
HomeActivity	Handles the logic behind the UI of the Home Activity.	WeekViewCalendar cal, List<string> assignments

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user has added an assignment to a class.	10/16/17			

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency:	Minimum: 0		Maximum:	Average:1	(OR)Fixed:
Per:	Hour: □	Day: ☒	Week: □	Month: □	Other:

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
N/A							

Volume Information

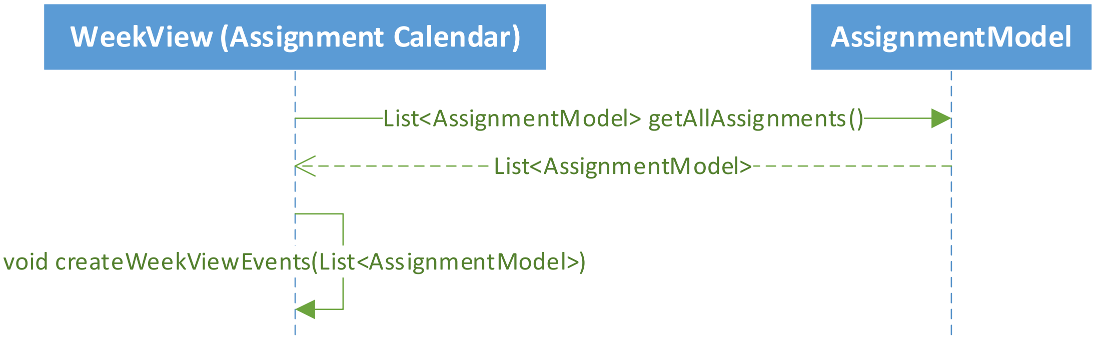
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 24 VAIAC Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
2. The system displays the Assignment Calendar on the Home Activity.
    - The WeekView class calls the AssignmentModel's getAllAssignments() method to get a List<AssignmentModel> (Note: The WeekView class is a third party class).
    - The WeekView class then calls its createWeekViewEvents(List<AssignmentModel>) method to add the List<AssignmentModel> as events in the WeekView calendar.
3. The Assignment Calendar functions as described in requirement # 3.2.1.
4. The user can see the assignments in the assignment calendar.

UC 24 VAIAC Sequence Diagram 01


WeekView (Assignment Calendar)
AssignmentModel
List<AssignmentModel> getAllAssignments()
List<AssignmentModel>
void createWeekViewEvents(List<AssignmentModel>)

UC 25

General Information	
Use Case Name\Number: TAIAC 01 Subject Area: Tapping an assignment in the Assignment Calendar. Description: The user taps on an assignment in the Assignment Calendar.	Responsible Analyst: Matthew Del Fante

Requirements/Feature Trace	
REQ#	**Requirements Name and / or Short Description**
3.2.1.4	Redirects user to View Assignments Activity when an assignment is tapped on in the Assignment Calendar.

Revision History		
Author	**Date**	**Comments**
Matthew Del Fante	10/16/17	Initial draft.

Insertion Points in other Use Cases		
Use Case Name	**Use Case Number**	**Step Inserted After**
N/A		

Actors		
Actor Name	**Person/System**	**Brief Description**
See Use Case Summary.		

Pre-Conditions	
#	Description
	See Use Case Summary.

Start Stimulus
The user taps the logo of the Class Hub app on his/her android device.

Use Case Main Course Steps			
Number	**Description**	**Adds/Alt Name/Number**	**Bus Rule#**
1	The system takes the user to the Class Hub Home Activity.		
2	The system displays the Assignment Calendar on the Home Activity.		
3	The Assignment Calendar functions as described in requirement # 3.2.1.		
4	The user taps on an assignment in the Assignment Calendar.		
5	The system redirects the user to the View Assignments Activity for the class that assignment is associated with.		

Exception Conditions		
Exception Situations	**Action(s) on Exception**	**Adds\Alt UC #**
The user hasn’t added an assignment to a class.	The user won’t see any assignments in the Assignment Calendar.	

Post-Conditions

#	Description
1	The system redirects the user to the View Assignments Activity for the class that the assignment is associated with.

Candidate Objects

Class/Object Name	Descriptions	Possible attributes
HomeActivity	Handles the logic behind the UI of the Home Activity.	WeekViewCalendar cal, List<string> assignments

Assumptions

#	Assumption	Date Raised	Raised By	Date Verified	Verified By
1	The user has added an assignment to a class.	10/16/17			

Issues

#	Issue	Date Raised	Raised By	Date Verified	Verified By
N/A					

Other Comments

Author	Comment	Date
N/A		

Frequency of Execution

Frequency: Minimum: 0 Maximum: Average: 1 (OR)Fixed:

Per: Hour: □ Day: □ Week: ☒ Month: □ Other:

Timing Information

#	At/ Between	Step(s)	Timing Unit	Minimum	Average	Maximum	Comments
N/A							

Volume Information

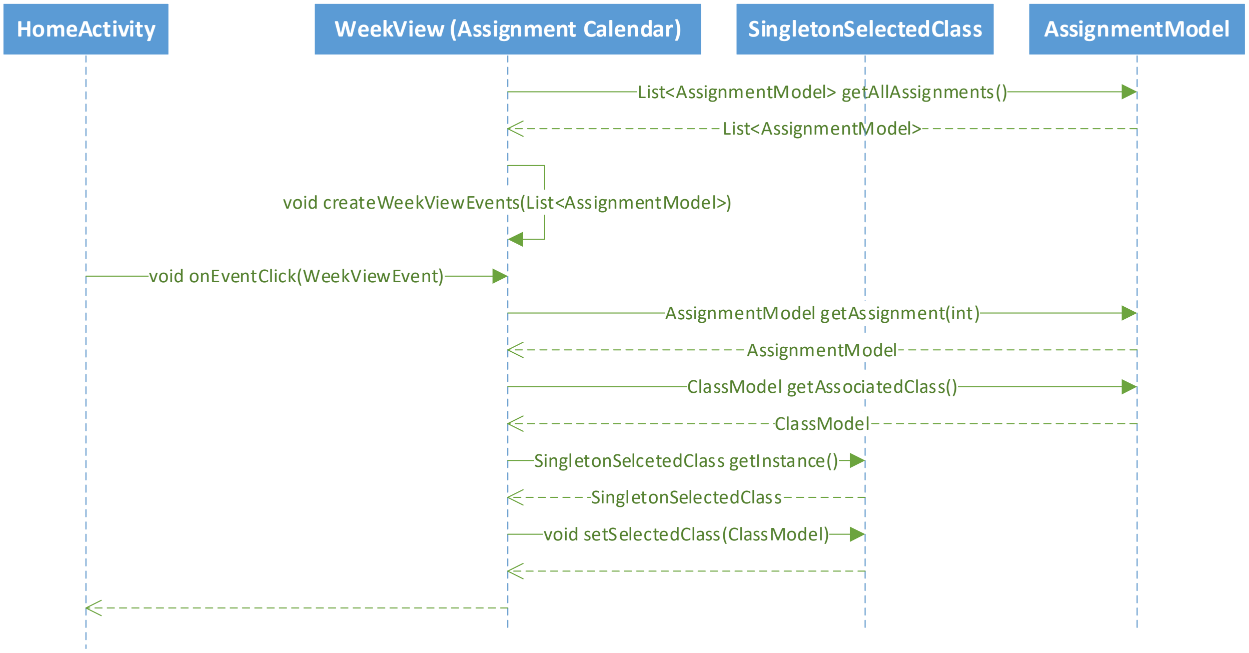
#	Step #	Unit of Measure	Minimum	Average	Maximum	Comments
N/A						

UC 25 TAIAC Scenario 01

Use Case Steps

1. The system takes the user to the Class Hub Home Activity.
2. The system displays the Assignment Calendar on the Home Activity.
   - The WeekView class calls the AssignmentModel's getAllAssignments() method to get a List<AssignmentModel> (Note: The WeekView class is a third party class).
   - The WeekView class then calls its createWeekViewEvents(List<AssignmentModel>) method to add the List<AssignmentModel> as events in the WeekView calendar.
3. The Assignment Calendar functions as described in requirement # 3.2.1.
4. The user taps on an assignment in the Assignment Calendar.
   - The HomeActivity triggers the WeekView's onEventClick(WeekViewEvent) method to be called.
   - The WeekView class then calls the AssignmentModel's getAssignment(int) method passing it the id of the WeekViewEvent that was tapped. The AssignmentModel that the getAssingment(int) method returns is the assignment that corresponded to the WeekViewEvent that was tapped. WeekView then calls the AssignmentModel's getAssociatedClass() method.
   - WeekView then calls the SingletonSelectedClass's getInstance() method followed by the SingletonSelectedClass's setSelectedClass(ClassModel) method passing the return value of the getAssociatedClass() method as a parameter to the method.
5. The system redirects the user to the View Assignments Activity for the class that assignment is associated with.

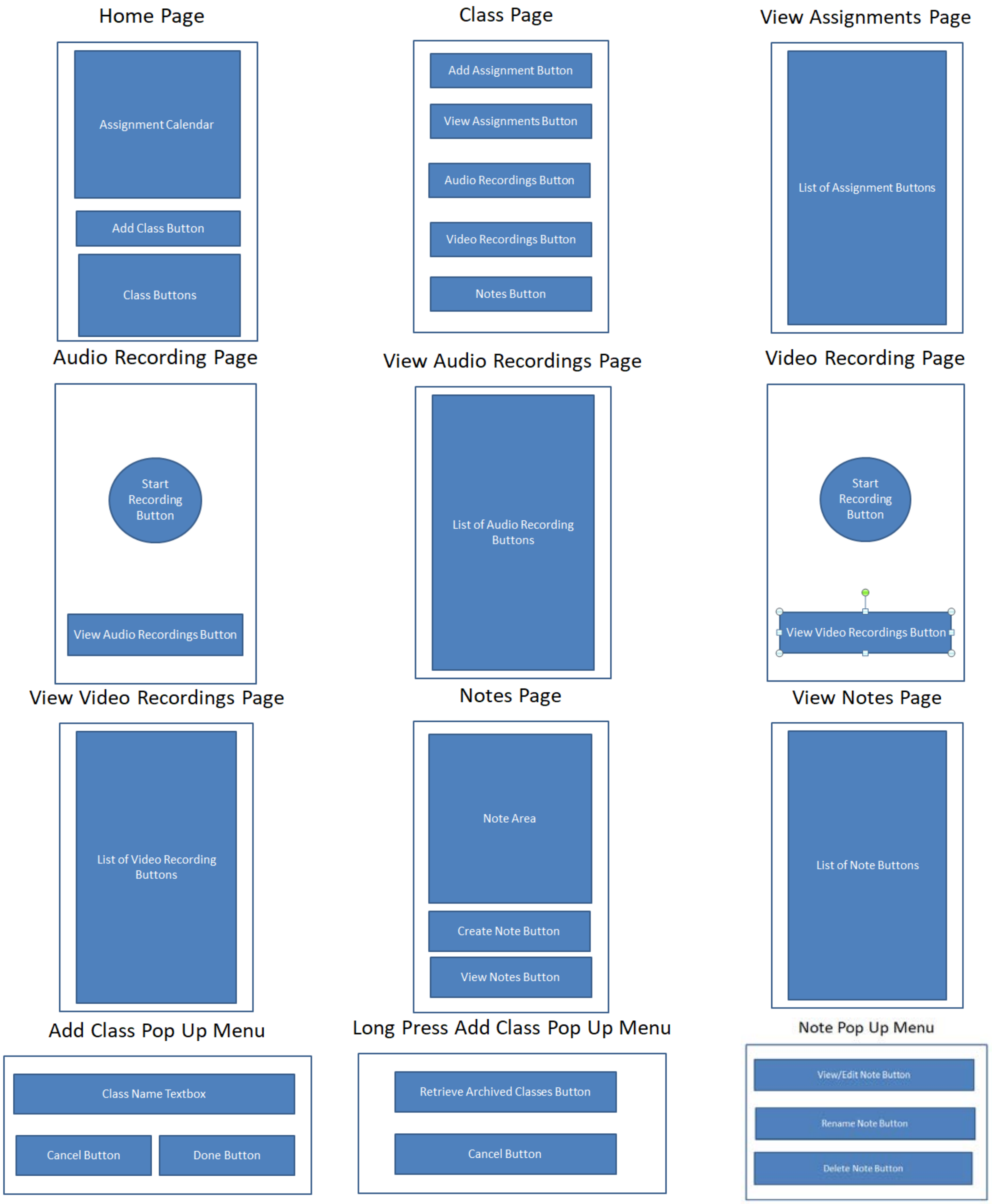
UC 25 TAIAC Sequence Diagram 01


HomeActivity
WeekView (Assignment Calendar)
SingletonSelectedClass
AssignmentModel
List<AssignmentModel> getAllAssignments()
List<AssignmentModel>
void createWeekViewEvents(List<AssignmentModel>)
void onEventClick(WeekViewEvent)
AssignmentModel getAssignment(int)
AssignmentModel
ClassModel getAssociatedClass()
ClassModel
SingletonSelcetedClass getInstance()
SingletonSelectedClass
void setSelectedClass(ClassModel)

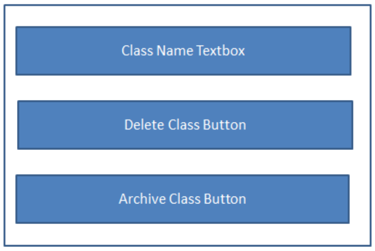
CRUD Matrix

Use Case ID	Use Case Name	Assignment Model	AudioRecording Model	Class Model	Note Model	VideoRecording Model	SingletonSelected Class
UC 01	Add Assignment 01	C		U			U
UC 02	Add Audio Recording 01		C	U			U
UC 03	Add Class 01			C			
UC 04	Add Notes 01			U	C		U
UC 05	Add Video Recording 01			U		C	U
UC 06	Archive Class 01			U			
UC 07	Delete Assignment 01	D		U			U
UC 08	Delete Audio Recording 01		D	U			U
UC 09	Delete Class 01			D			
UC 10	Delete Note 01			U	D		U
UC 11	Delete Video Recording 01			U		D	U
UC 12	Edit Assignment 01	U		U			U
UC 13	Edit Notes 01			U	U		U
UC 14	Listen to Audio 01		R	R			UR
UC 15	MAAC 01	U		U			U
UC 16	Rename Audio Recording 01		U	U			U
UC 17	Rename Class 01			U			
UC 18	Rename Note 01			U	U		U
UC 19	Rename Video Recording 01			U		U	U
UC 20	Retrieve Archived Classes 01			R			
UC 21	View Assignment 01	R		R			UR
UC 22	View Notes 01			R	R		UR
UC 23	Watch Video Recording 01			R		R	UR
UC 24	VAIAC 01	R		R			
UC 25	TAIAC 01	R		R			

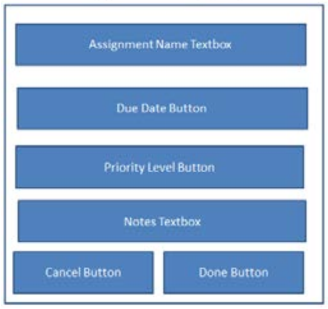
Low Fidelity UI



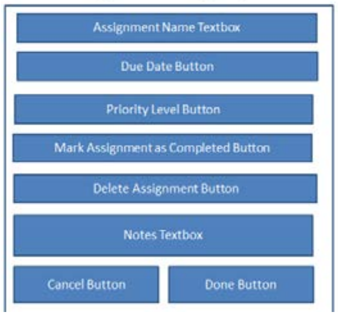
Long Press Class Pop Up Menu



Add Assignment Pop Up Menu



Edit Assignment Pop Up Menu


Assignment Name Textbox
Due Date Button
Priority Level Button
Mark Assignment as Completed Button
Delete Assignment Button
Notes Textbox
Cancel Button
Done Button