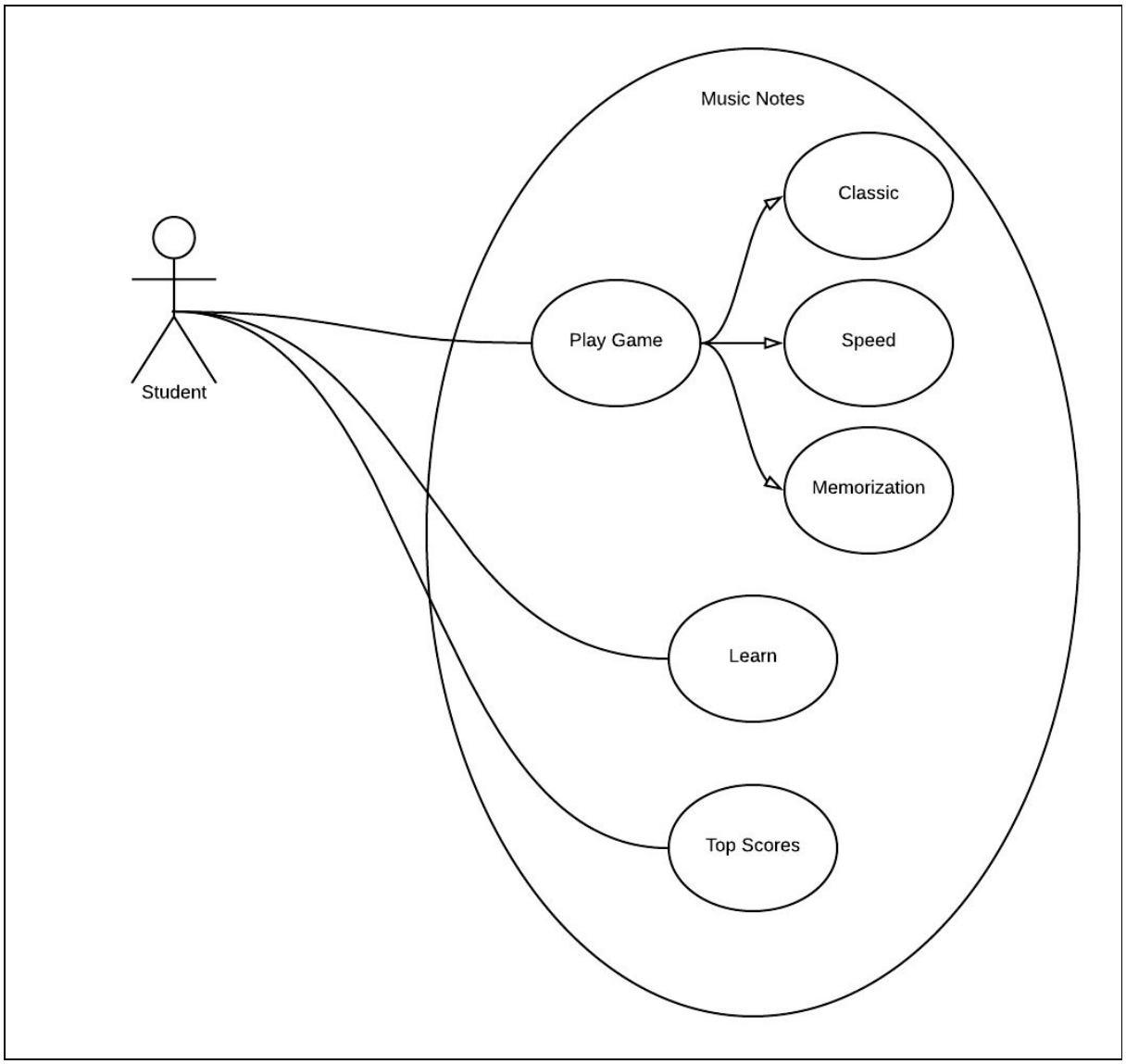
Anthony Russo
Music Notes
CSC - 415
Github: https://github.com/russo-anthony/Assignment3_Russo
VM: student1@csc415-server43.hpc.tcnj.edu

OSS Analysis and Design - Assignment 4

Use Case Diagram:



Use Case Descriptions:

Use Case: Play Classic Mode
Primary Actor: Student
Goal In Context: Play Music Notes in “Classic” gamemode
Precondition: Student selects “Play” from home menu
Trigger: Student decides to play the game, and chooses “Classic” play style
Scenario: 1. Student decides to play Music Notes 2. Student opens app and selects “Play” from home menu 3. Student then selects “Classic” gamemode 4. Student plays Music Notes in Classic mode until satisfied 5. Student then exits back to the home menu or chooses another play style
Exceptions: 1. Student chooses another gamemode
Priority: High, main component of the app
When Available: First development objective
Frequency of Use: Frequent
Channel to Actor: Access to smartphone, and the internet
Secondary Actors: None
Open Issues: Effective and simple UI design

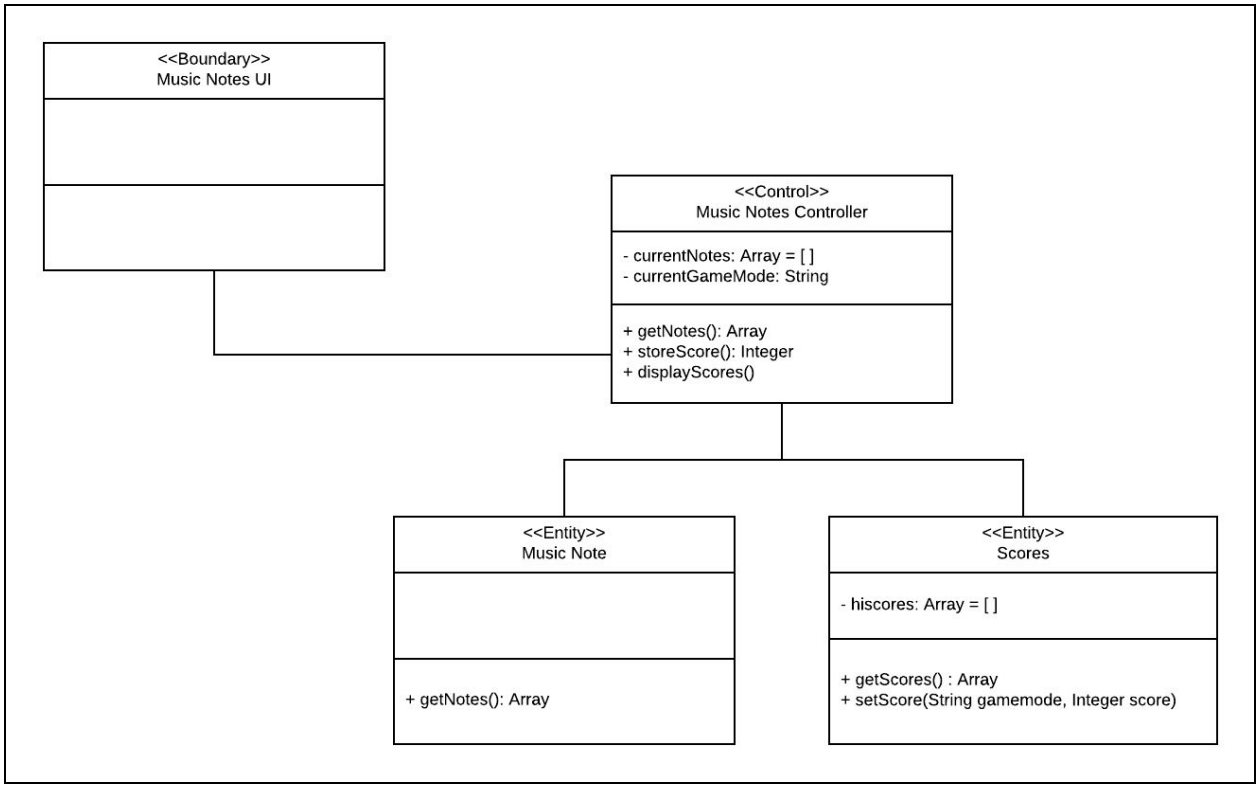
Use Case: Play Speed Mode
Primary Actor: Student
Goal In Context: Play Music Notes in “Speed” gamemode
Precondition: Student selects “Play” from home menu
Trigger: Student decides to play the game, and chooses “Speed” play style
Scenario: 1. Student decides to play Music Notes 2. Student opens app and selects “Play” from home menu 3. Student then selects “Speed” gamemode 4. Student plays Music Notes in Speed mode until satisfied 5. Student then exits back to the home menu or chooses another play style
Exceptions: 1. Student chooses another gamemode
Priority: Medium , main component of the app, but not essential
When Available: Second development objective
Frequency of Use: Frequent
Channel to Actor: Access to smartphone, and the internet
Secondary Actors: None
Open Issues: Effective and simple UI design

Use Case: Play Memorization Mode
Primary Actor: Student
Goal In Context: Play Music Notes in “Memorization” gamemode
Precondition: Student selects “Play” from home menu
Trigger: Student decides to play the game, and chooses “Memorization” play style
Scenario: 1. Student decides to play Music Notes 2. Student opens app and selects “Play” from home menu 3. Student then selects “Memorization” gamemode 4. Student plays Music Notes in Memorization mode until satisfied 5. Student then exits back to the home menu or chooses another play style
Exceptions: 1. Student chooses another gamemode
Priority: Medium, main component of the app, but not essential
When Available: Third development objective
Frequency of Use: Frequent
Channel to Actor: Access to smartphone, and the internet
Secondary Actors: None
Open Issues: Effective and simple UI design

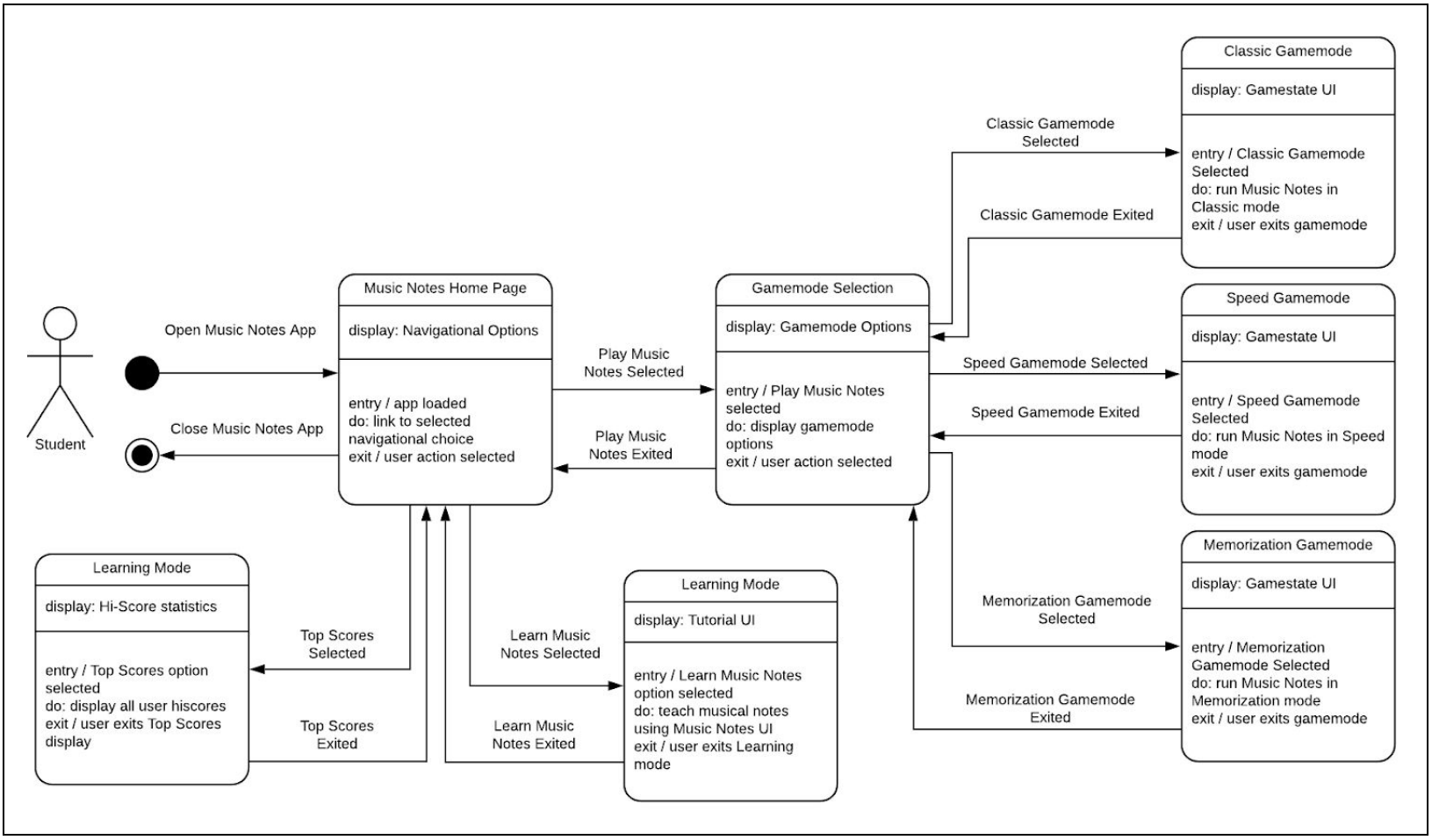
Use Case: Utilize Learning Feature
Primary Actor: Student
Goal In Context: Teach the standard seven musical notes
Precondition: Student selects “Learn” from home menu
Trigger: Student decides to learn about how to read the different music notes
Scenario: 1. Student decides to learn about music 2. Student opens app and selects “Learn” from home menu 3. Student used the interactive UI to learn the different music notes 4. The student can continue to test their knowledge in Learning mode or exit to play the other game modes present in the app
Exceptions: 1. Student chooses to play the other game modes instead of learning
Priority: High, main component of the app
When Available: First development objective
Frequency of Use: Frequent
Channel to Actor: Access to smartphone, and the internet
Secondary Actors: None
Open Issues: Effective and simple UI design

Use Case: View Top Scores
Primary Actor: Student
Goal In Context: Display hiscores the user has achieved
Precondition: Student selects “Top Scores” from home menu
Trigger: Student decides to view hiscores
Scenario: 1. Student decides to view hiscores 2. Student clicks “Top Scores” in home menu 3. Hiscores are displayed 4. The Student then has the option to exit back to the home menu
Exceptions: 1. Student chooses another option to Play or Learn
Priority: Low, main component of the app, but not essential
When Available: Third development objective
Frequency of Use: Moderate
Channel to Actor: Access to smartphone, and the internet
Secondary Actors: None
Open Issues: Effective and simple UI design

Detailed Design Class Diagram:

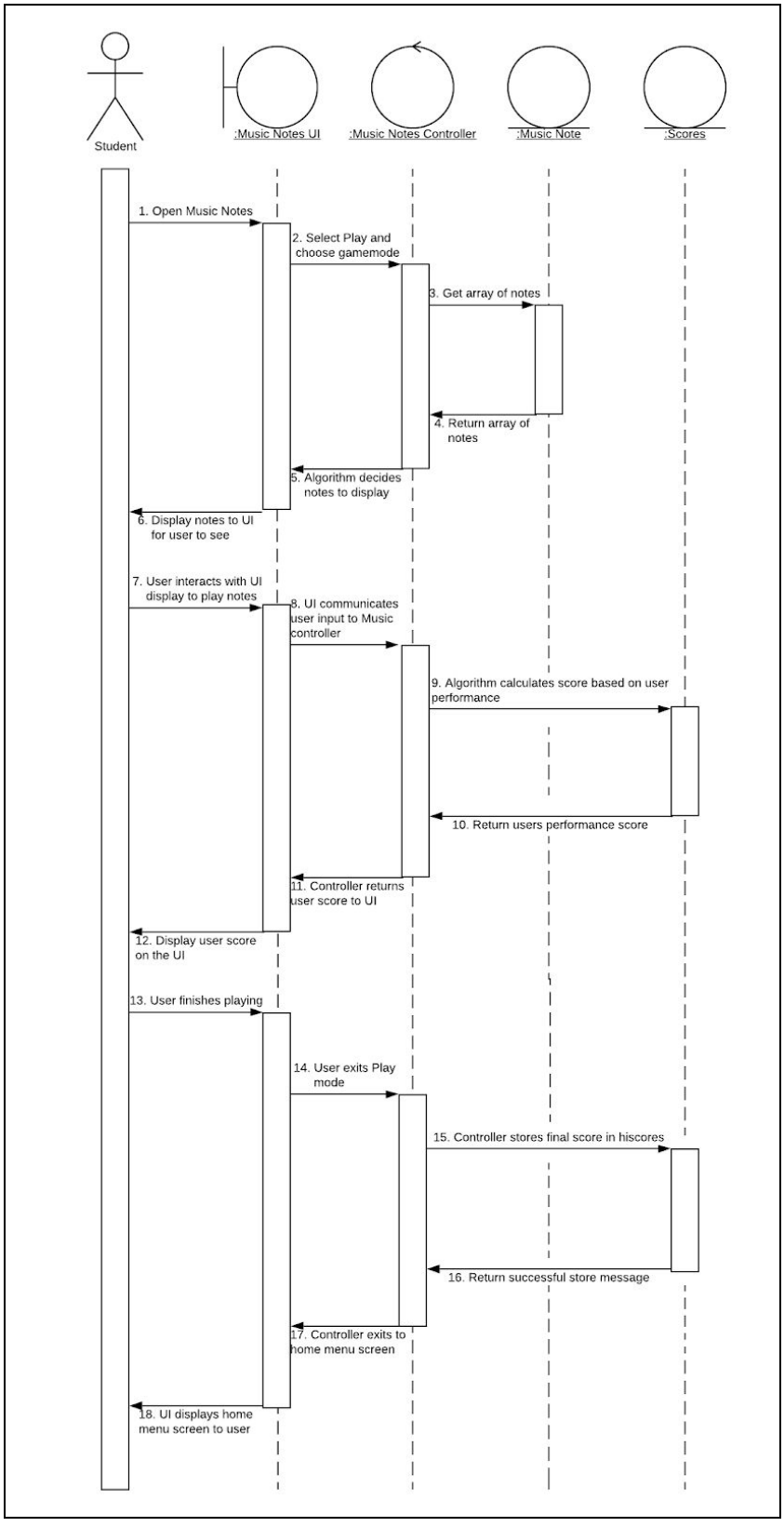


Statechart:

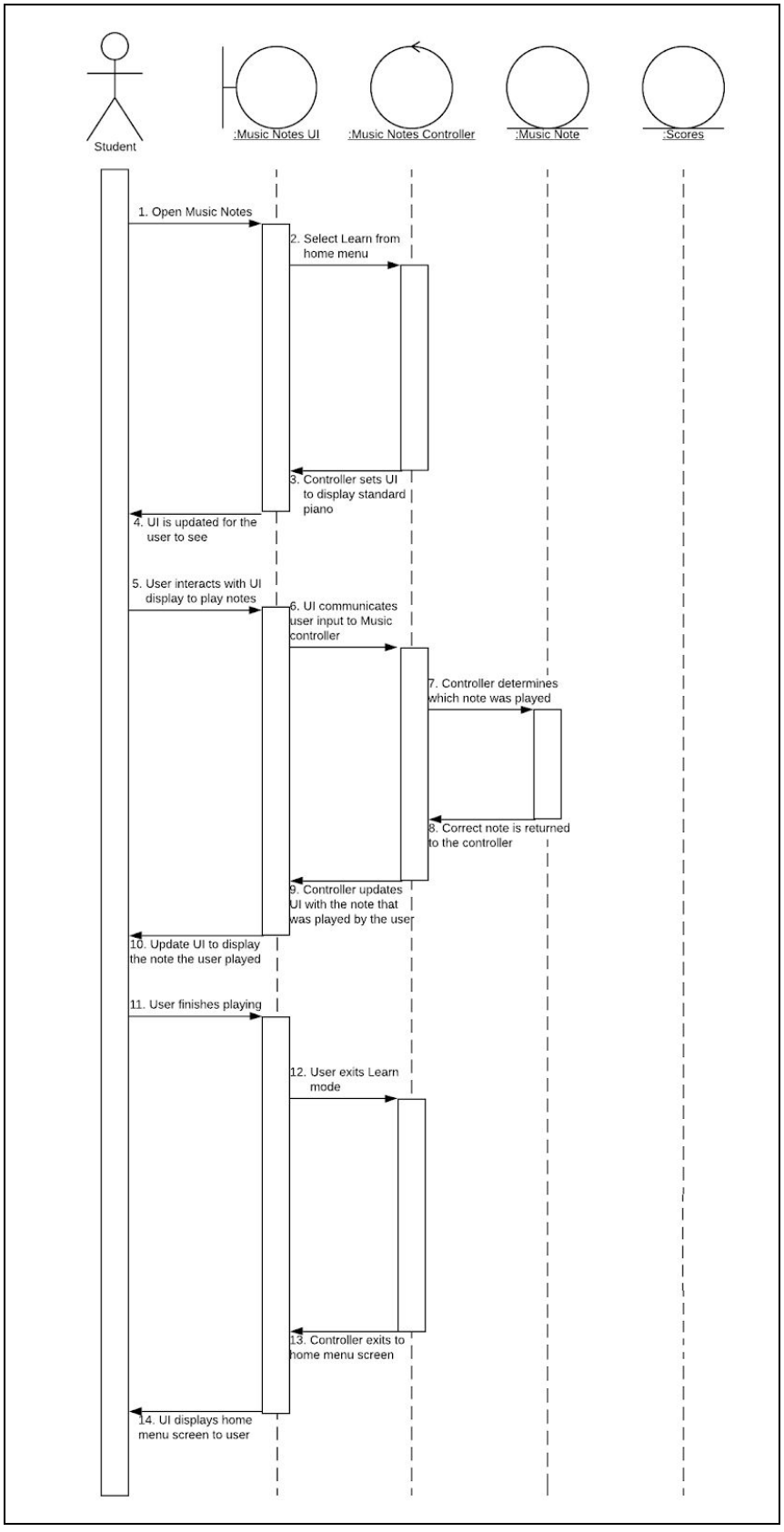

Student
Open Music Notes App
Close Music Notes App
Music Notes Home Page
display: Navigational Options
entry / app loaded
do: link to selected navigational choice
exit / user action selected
Play Music Notes Selected
Play Music Notes Exited
Gamemode Selection
display: Gamemode Options
entry / Play Music Notes selected
do: display gamemode options
exit / user action selected
Classic Gamemode
display: Gamestate UI
entry / Classic Gamemode Selected
do: run Music Notes in Classic mode
exit / user exits gamemode
Classic Gamemode Selected
Classic Gamemode Exited
Speed Gamemode
display: Gamestate UI
entry / Speed Gamemode Selected
do: run Music Notes in Speed mode
exit / user exits gamemode
Speed Gamemode Selected
Speed Gamemode Exited
Memorization Gamemode
display: Gamestate UI
entry / Memorization Gamemode Selected
do: run Music Notes in Memorization mode
exit / user exits gamemode
Memorization Gamemode Selected
Memorization Gamemode Exited
Learning Mode
display: Hi-Score statistics
entry / Top Scores option selected
do: display all user hiscores
exit / user exits Top Scores display
Top Scores Selected
Top Scores Exited
Learning Mode
display: Tutorial UI
entry / Learn Music Notes option selected
do: teach musical notes using Music Notes UI
exit / user exits Learning mode
Learn Music Notes Selected
Learn Music Notes Exited

System Sequence Diagram:

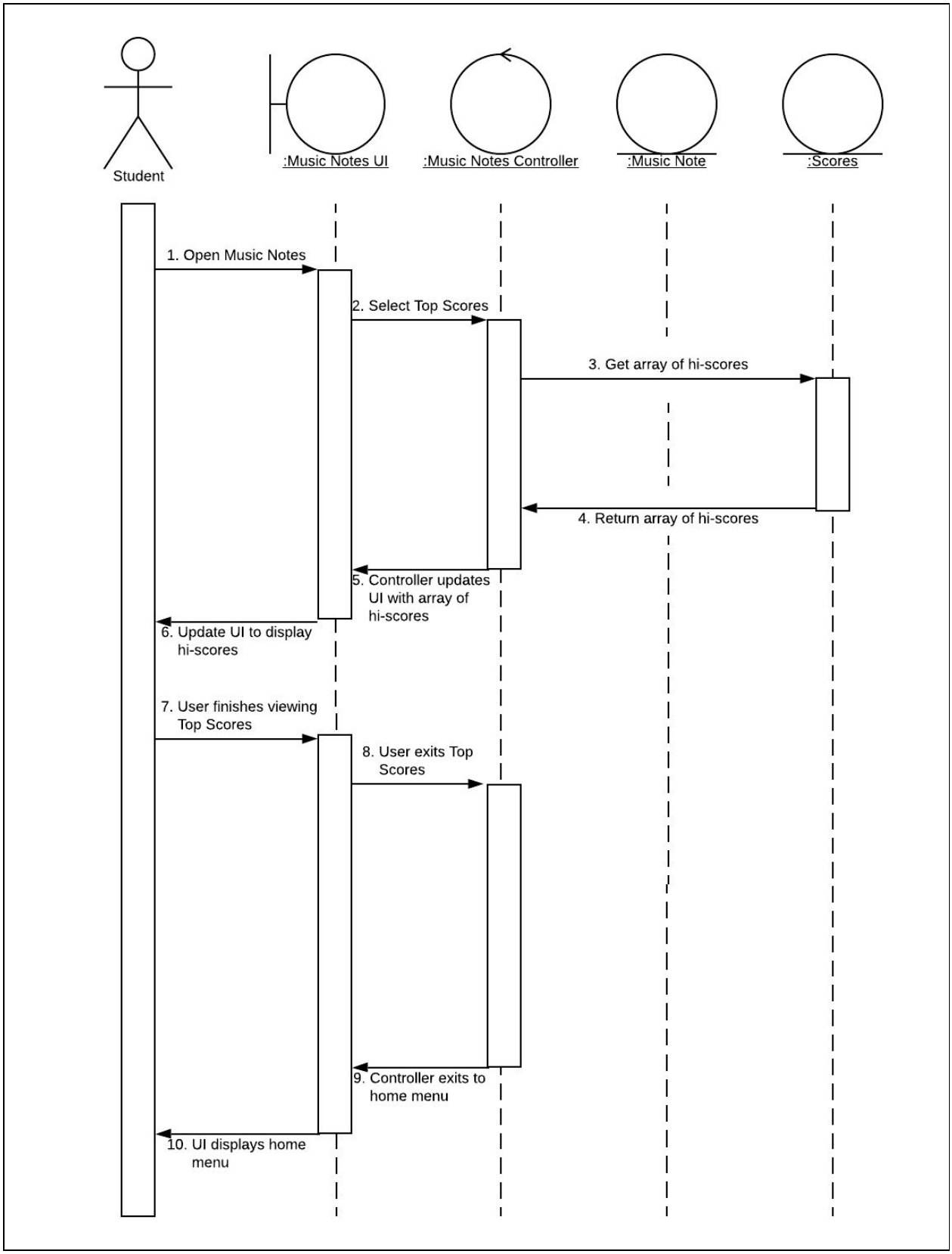
- **Play Music Notes**



- **Learn Music Notes**

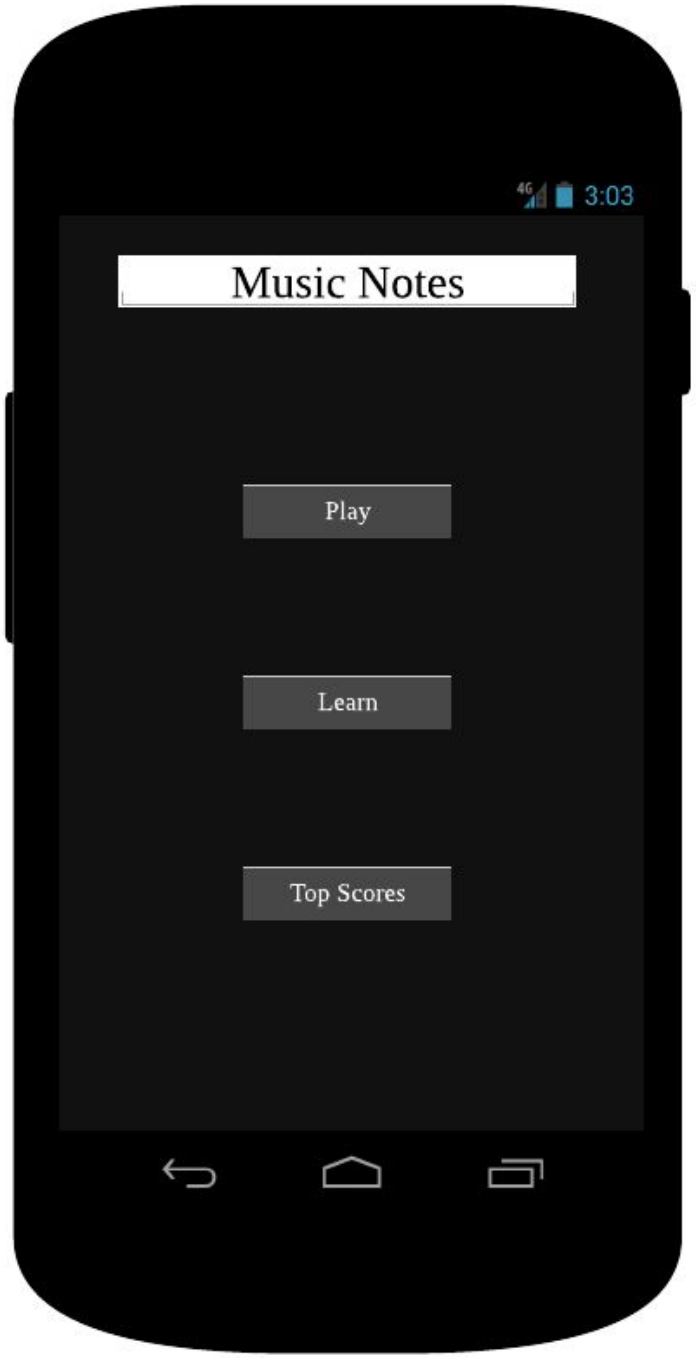


- **Top Scores**

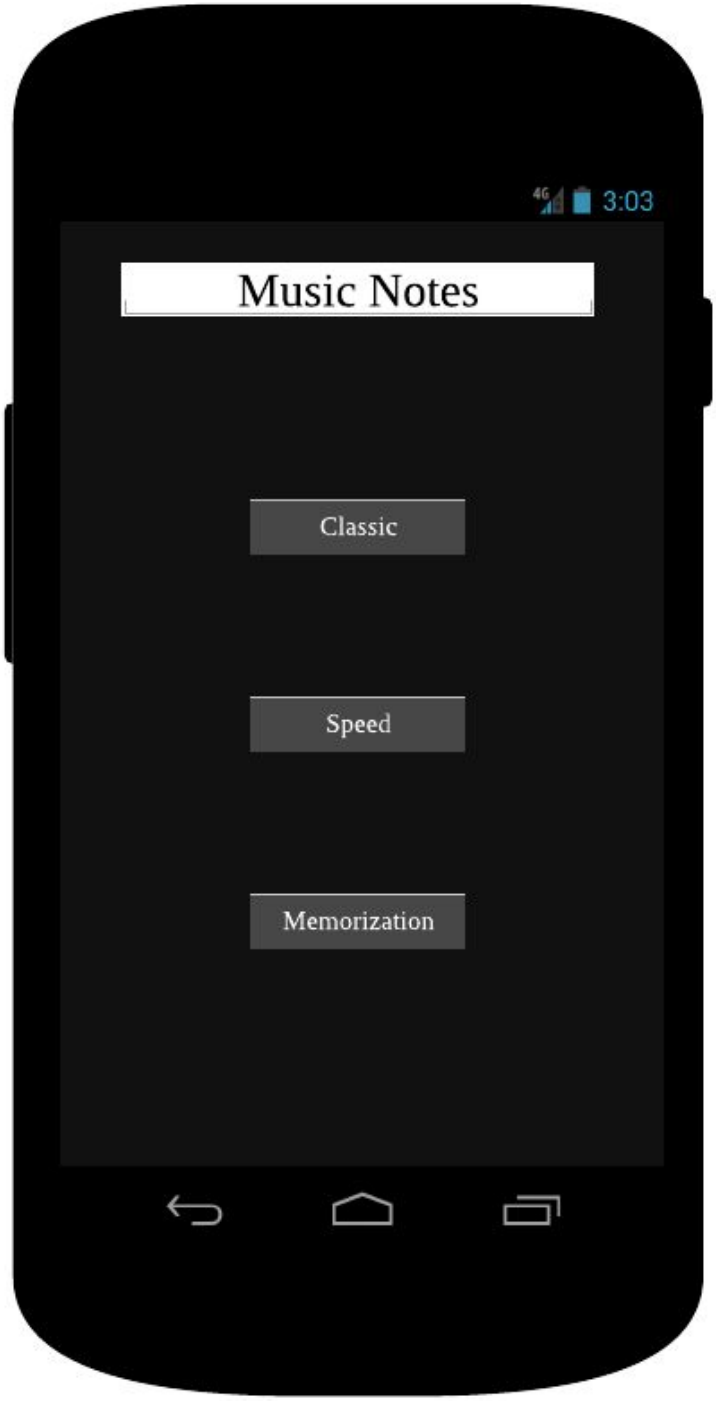

Student
:Music Notes UI
:Music Notes Controller
:Music Note
:Scores
1. Open Music Notes
2. Select Top Scores
3. Get array of hi-scores
4. Return array of hi-scores
5. Controller updates UI with array of hi-scores
6. Update UI to display hi-scores
7. User finishes viewing Top Scores
8. User exits Top Scores
9. Controller exits to home menu
10. UI displays home menu

Mock Ups:

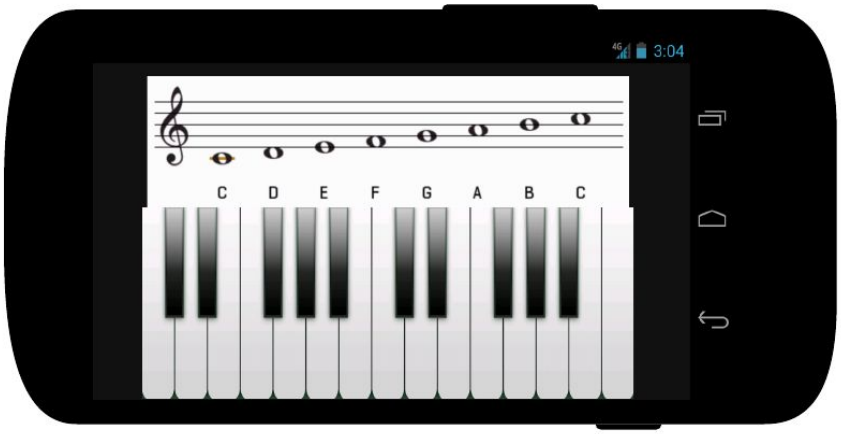
- **Home Menu**



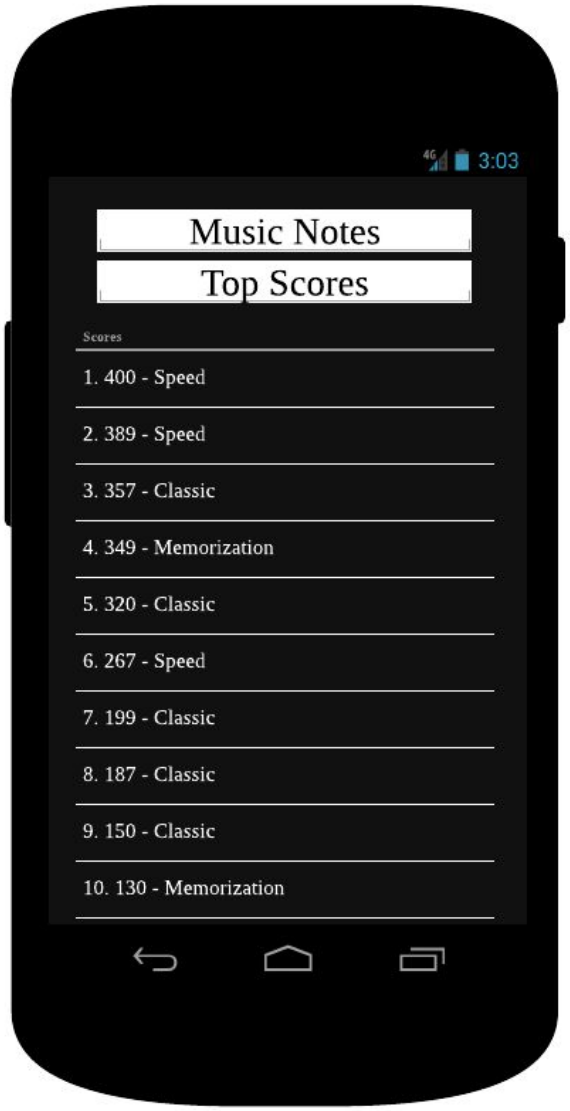
- **Play Menu**



- **In-Game UI**



- **Top Scores**



Principles of UI Design:

1. Strive for Consistency
    a. Buttons are placed in the same areas with a general hierarchy of use. Color schemes are consistent to keep the design clean, and the UI is clean with little extras to confuse the user.
2. Enable Frequent Users to Use Shortcuts
    a. Most of the menus only require a few steps to reach so there is no need for shortcuts in the UI. The app is generally straightforward and access is quick.
3. Offer Informative Feedback
    a. Menus and buttons are all labeled and updated as the user interacts with the UI to ensure the user understands what they are doing and what is happening within the app.
4. Design Dialogs to Yield Closure
    a. Success messages and clear exits will ensure the user that the app is functioning correctly and they are clear about what is happening.
5. Offer Simple Error Handling
    a. There is little to no user input involved besides button presses, and interacting with the keyboard and notes so error handling will be generally uncomplicated. Error handling will be mostly internal so ensure the correct outputs are displayed to the screen.
6. Permit Easy Reversal of Actions
    a. The android back button will always be accessible and will be the main component in reversing a selection or exiting a menu.
7. Support Internal Locus of Control
    a. The UI of the app is very minimalistic in design, allowing the user to have a simple view of what to do with little confusion about how to control the app. Every choice will be up to the user, and they will have full control over the functionality and use of the app.
8. Reduce Short-Term Memory Load
    a. The note sounds and graphics are very simple and should not take up too much space, however they will be loaded on an as needed basis to limit the use of memory.

Modularity and Encapsulation:

The app is split into its main components such as the UI, controller class, and the resources and storage of scores, note sounds, and graphics. By splitting these up the app design is modular and has the ability to expand, by possibly adding the implementation of other instruments to the app. This also keeps the score information in a separate object class hidden from the main control section and the UI modules of the app.

Elegance and Efficiency of the Algorithms:

The internal algorithms of the app will be simple and implement the functionality of the app in a very straightforward manor to be as efficient as possible. The loading, choosing, matching, and display of the notes on the display will be implemented such that the control class can choose the notes and display methods based on the gamemode that the user is playing in. For each different gamemode there will be some different display techniques when choosing the notes and the speed at which they are displayed, but the loading and matching of the notes will be the same to modularize the code as best as possible and to keep the length of the code down. Since music is supposed to flow nicely it is imperative that the algorithms are quick enough to not interrupt the flow of the music, and so that the user has an enjoyable experience using the app.

Appropriate Data Structures:

The data structures I will be implementing in this app will not be very complicated since the data I am storing is not very complex. The user's hi-scores will simply be stored in an array and sorted by descending order to show the highest scores first on the list. The notes will also be stored in an array and sorted in an order to represent a certain song, or in a random pattern depending on the gamemode that the user chooses to play.

Test Case Design:

For unit testing, I will simply test each function of the app to ensure that I am getting all of the expected results that I should be seeing. Once each portion of the app is tested and proven to be working successfully by itself, it can then be integrated into the overall design of the app. Once each module of the app is integrated tests will be done again to ensure that the new module works fluidly with the other existing components. This way I will be able to catch any bugs that may present themselves as I am working rather than finding them later in the final testing stages. Once the system is fully assembled there will be final tests to make sure there are no final bugs and to make sure the system is efficient, easy to use, and is appealing. Finally I will be able to get feedback on the effectiveness of the app from many of my friends who are Music Education majors. They will be able to give me feedback about how well the app would be able to educate kids about music, and I can tweak the design to improve it based on their feedback. To debug any problems I may be having during testing and integration I will be using the Android Studio debugging tools. It is a robust system and allows you to set breakpoints, and view current variable values throughout your code which will be able to help me find where my mistake are.

Functionality Tested	Inputs	Expected Outputs	Actual Outputs
Play	Click on Play	Display gamemode selection	
Learn	Click on Learn	Display Piano UI, correct notes play when pressed	
Top Scores	Click on Top Scores	Display hi-scores list	
Classic Mode	Click on Classic Mode	Display Piano UI, correct notes play when pressed, playthrough is smooth	
Speed Mode	Click on Speed Mode	Display Piano UI, correct notes play when pressed, playthrough speed is increased over time	
Memorization Mode	Click on Memorization Mode	Display Piano UI, correct notes play when pressed, notes appear then disappear	
Back Button	Click on the Back Button	Return to previous menu screen	
A Note	Press the A Key	Play the A note sound clip	
B Note	Press the B Key	Play the B note sound clip	
C Note	Press the C Key	Play the C note sound clip	
D Note	Press the D Key	Play the D note sound clip	
E Note	Press the E Key	Play the E note sound clip	
F Note	Press the F Key	Play the F note sound clip	
G Note	Press the G Key	Play the G note sound clip	

Open Source Maintenance and Communication:

Open source projects are difficult to maintain sometimes because it takes a team of dedicated developers to continually work of a project to improve it over time. Many of these open source projects are worked on by people who are not always paid to develop them or it is not always their main focus so sometimes projects can become unattended. However there are still a large amount of open source projects that flourish because of the work that people put into them, and many of these people stay with the project throughout its growth. Many of these dedicated members of the project are usually the people who over time become the lead developers and senior staff that oversee much of the management and maintenance of the project. In many of these open source project communities, developers will be able to communicate on forums or other online sources to try and contribute the the project. These open source projects generally have a strict contribution policy to ensure that stuff that is being developed and committed to the project is quality and worthwhile work to be considered being added to the project. The lead developers and senior project members with the most experience are usually the ones who would have the final say to a new addition, and are most likely the ones to make sure the code is quality work. Sites like Github are used to organize all of the projects code, mark down issues and bugs, and also communicate goals and objectives for the project.

For the Music Notes Open Source project I will be implementing a similar system to encourage and manage outside contributions to the project. The Github repository for Music Notes will be a place where community members can learn how they can contribute, and can fork their own version of the code in order to work on it. They can they submit their revisions to be checked and possibly added to the existing project. Future goals and objectives can also be addressed using the Milestones feature that Github has. Issues and bugs found in the code can also be addressed using Githubs Issues tab in the repository. In order for someone to contribute code to the project their code must be clean, well written, documented and easily understood. They must also test the code thoroughly before submitting it for review and include their unit test to be used in the code reviewing process.

Open Source Licensing:

The current top open source licenses include the MIT license, the Apache 2.0 license, and the GPL 3.0 license. The MIT license has been gaining popularity because it allows others to use your work however they like, as long as they provide you with the credit for the work and don't hold you liable for the use of your project. This license is good because it basically protects you from anything that could go wrong with someone else using your work. The Apache license seems to be similar to the MIT license but requires a little more licensing and patent cooperation and that modifications and works including your code be distributed without the source code. This license seems to help protect your work a little better than the MIT license does. The last license is the GPL license which is a very open and free license with key features such as the freedom to use, modify and share the code however you like. I think the GPL

license would make the most sense for the Music Notes project because using the GPL license will give the project the ability to share, change, expand and grow on a much bigger scale. Doing this will allow the project to reach a larger audience and affect more people's lives. Since the main objective of the project is education, this seems like a good idea because the app will be able to educate more students with little issues.