Matt Zidar
Assignment 3 Deliverables

Pathname to Project on VM: sysadmin@csc415@server20.hpc.tcnj.edu. Located in /webapp/MusicApp.
Pathname to Private Repo on Github: https://github.com/mattzidar/MusicalStudentsGroup

1. Use Case Descriptions -

Use Case - Create of edit donation listing

Primary Actor - Student with extra instruments, teacher, alumni

Goal in Context - Create a free listing for a donation of an instrument that students in need can request and get. The creator of the donation listing can delete it or edit it by using their private user ID they created when they made the listing.

Preconditions - User must have access to the application and must fill out every field for information about the donation that they will be providing. Deleting or editing their donation listing requires that their listing already exists and that the user has access to their user ID which they created upon making the donation listing.

Trigger - User wants to donate an instrument, they click the 'create donation listing' button in the donations tab. If the user wants to delete or edit their listing, they hit the 'delete' or 'edit' button in the donations tab.

Scenario -

- The user enters the application.
- The user selects the tab for 'Donations'.
- The user clicks 'Create a Donation Listing' button.
- The system displays fields for the user to enter all information associated with their listing.
- The user hits 'submit listing' and their donation listing is added to the system and sent to the database. The user is prompted that this was a success.
- The system displays their donation listing on the screen along with all the other donation listings open.
- User clicks on delete / edit listing and validates that action with their unique ID. User confirms deletion and listing is deleted from table and database. If the user selects to edit, they edit all necessary fields and submit the changes.
- If the user ID matches the one they created, the deletion / edits made to the listing are reflected by the system on the screen where all donation listings are shown.

Exception -

- User does not enter every field when creating a donation listing - System tells the user to go back and fill in the fields that they did not fill in.

- User ID they create is not a number up to 4 digits - the system tells them to create a new user ID that is a number up to four digits.
- The ID user enters when they edit or delete their listing does not match the ID they first made - system does not allow them to edit or delete that listing and requires that they enter the correct number.

Use Case - User requests donation item

Primary Actor - Student

Goal in Context - Allow users to search through listings based on criteria and select a donation item they would like to request if they need an instrument.

Preconditions - Users must have access to the application, donation items must exist in their area that is suitable for them.

Trigger - A student desires an instrument that they do not have, they click 'search for donation' or look through the entire list of donations in the 'donations' tab for an instrument that suits them. The student clicks 'request' and contacts the donor if they find a good listing.

Scenario -
- The user enters the application.
- The user clicks on the 'Donations' tab.
- The user clicks 'search for donation' if they wish to search more specifically or they search through the list for a donation item.
- The system displays results based on their search.
- The user clicks 'request' when they find a donation item that is suitable for them.
- The system returns the donor's contact information to the user.
- The user contacts the donor to receive the donation item.

Exceptions -
- The user search does not return any results - the user is prompted to refine their search to get a result that is suitable for them.

Use Case - Create or edit lesson listing for students

Primary Actor - Student, teacher

Goal In Context - Create a free listing for lessons on a particular instrument, mainly taught by older students but teachers in the district can provide these lessons as well. Lesson listing can be deleted as well.

Preconditions - User must have access to the application and must fill in every field for information about lessons that they will be providing. Deleting or editing information about listing requires that the user has access to the ID for the listing, and that the listing already exists.

Trigger - Student or teacher decides to offer lessons, they push a button to create this lesson listing in the lessons tab. Additionally, the student or teacher may desire to delete or edit their listing by hitting the 'delete' or 'edit listing' button in the lessons tab.

Scenario -

- The user enters the application.
- The user selects the tab for 'Lessons'.
- The user clicks the 'create a listing' button.
- The system displays fields for the user to enter all information associated with them and their lesson listing, including a field for a unique ID.
- The user enters the information in all fields.
- The user hits 'submit listing' and is prompted that their listing was added successfully by the system.
- The system displays their lesson listing in the list on the screen.
- Users can come back and delete / edit the listing with a unique ID created earlier by clicking the 'delete' or 'edit listing' buttons respectively.
- If user ID matches, the system tells the user that their listing has been deleted / edited successfully.

Exceptions -

- User does not enter information into all of the fields provided - the user is instructed to enter the missing information before submission.
- The Student attempts to enter something other than an integer (up to four numbers) for their private ID - The system tells them to re enter a number that is up to four digits.
- Unique ID does not match ID when attempting to delete or edit listing - the user gets two more tries but is locked out of deleting or editing the listing if all attempts are used.

Use Case - Request Lessons

Primary Actor - Student

Goal In Context - Allow students to look at a list of lessons or search based on criteria and request the lessons if a good match is found.

Preconditions - Lessons for that particular instrument in that area must exist for students to be able to request lessons.

Trigger - If a student desires to take lessons from an older student or teacher, they click ‘search for lesson’ or look through the entire list and request a lesson in the lessons tab. The student can put in a request for the lesson if they find a good person to teach them.

Scenario -

- Student enters the application.
- The student clicks on the tab for lessons.
- The student clicks on ‘search for lesson’ if they desire to search with specific criteria.
- The system displays matches based on their search.
- The student looks through matches based on search or looks through the entire list of lessons being offered.
- Upon finding a suitable person for lessons, the student clicks ‘request’.
- The system provides the student with information for contacting them.
- The student contacts the person for lessons.

Exceptions -

- Information in search does not return any results - the user is prompted to redefine their search from the system or to search the entire list.

Use Case - Create and update musical student group listing

Primary Actor - Student

Goal In Context - Allow for users to make a listing for a specific type of musical group they want to form in their district, information including style of music, instrument players needed, etc. Users can delete or edit this listing as new people make a request to join or if the request has been satisfied.

Preconditions - Students must have access to the application and must fill in every field related to information about the type of group they want to form. The listing must exist and the student must have their ID for the listing in order to edit or delete it.

Trigger - The student desires to form a musical group, they click the ‘Form a Group Listing’ button to create a musical group listing in the ‘Student Groups’ tab. After creating the listing, if the student desires to edit or delete this listing, they can click ‘edit listing’ or ‘delete listing’ to do so.

Scenario -

- The student enters the application.
- The student clicks on the ‘Student Groups’ tab.

- The student clicks on the 'Form a Group Listing' button if they want to create a musical group listing that other students can request to join.
- The system provides the student with fields where they enter all the information associated with the music group that they are trying to form, as well as contact information and unique ID.
- The student enters all information into the page.
- The student hits the 'Submit Group Listing' button.
- The system tells the user that their listing has been submitted successfully.
- The system adds their listing to the page which has each listing on it.
- The student can hit the 'edit' or 'delete' button in the same tab if they want to edit the information on their listing or delete it once it has been satisfied.
- If their ID matches that of the listing they created, their listing will be edited / deleted as specified.
- The system tells the user that their action has been completed.

Exceptions -

- The Student does not enter information into all of the fields when creating a listing - The system tells the user to go back and enter this information before submission
- The Student attempts to enter something other than an integer (up to four numbers) for their private ID - The system tells them to re enter a number that is up to four digits.
- The ID that the student enters does not match the ID of the listing when trying to edit or delete it - They are given two more tries but are locked from attempting to edit or delete the listing if they use all of those attempts.

Use Case - Request to join student music group

Primary Actor - Student

Goal In Context - Allow the student a method of searching through all of the music group listings, or the student can search based on criteria they specify. Upon finding a match that is good for them, the student can put in a request to join this music group and they are provided information to contact the student who put up the listing.

Preconditions - A group listing must already exist in the area if a student wants to request to join it.

Trigger - If a student desires to join a musical group from other students in their area, they click 'Search for a Group' if they want to search for one based on criteria or they can search the entire list in the 'Student Groups' tab. A request can be put in when they find a suitable group.

Scenario -

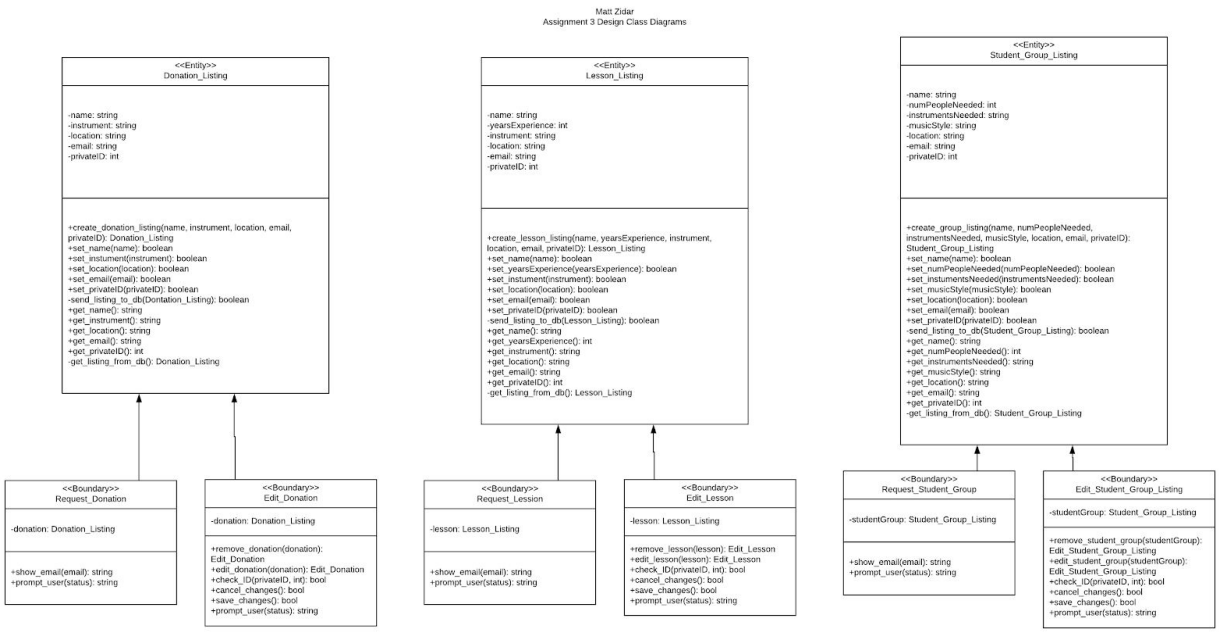
- The student enters the application.

- The student clicks on the ‘Student Groups’ tab.
- The student clicks on the ‘Search for a Group’ button if they want to search for a musical group to join based on their criteria.
- The system displays matches based on their search.
- The student looks through the entire list of group listings, or searches through the matches provided to them based on their search.
- If a good group has been identified by the student, they click the ‘Request’ button.
- The system provides the user all information necessary to contact the person who made the listing.
- The student contacts the other student about joining their group.

Exceptions -

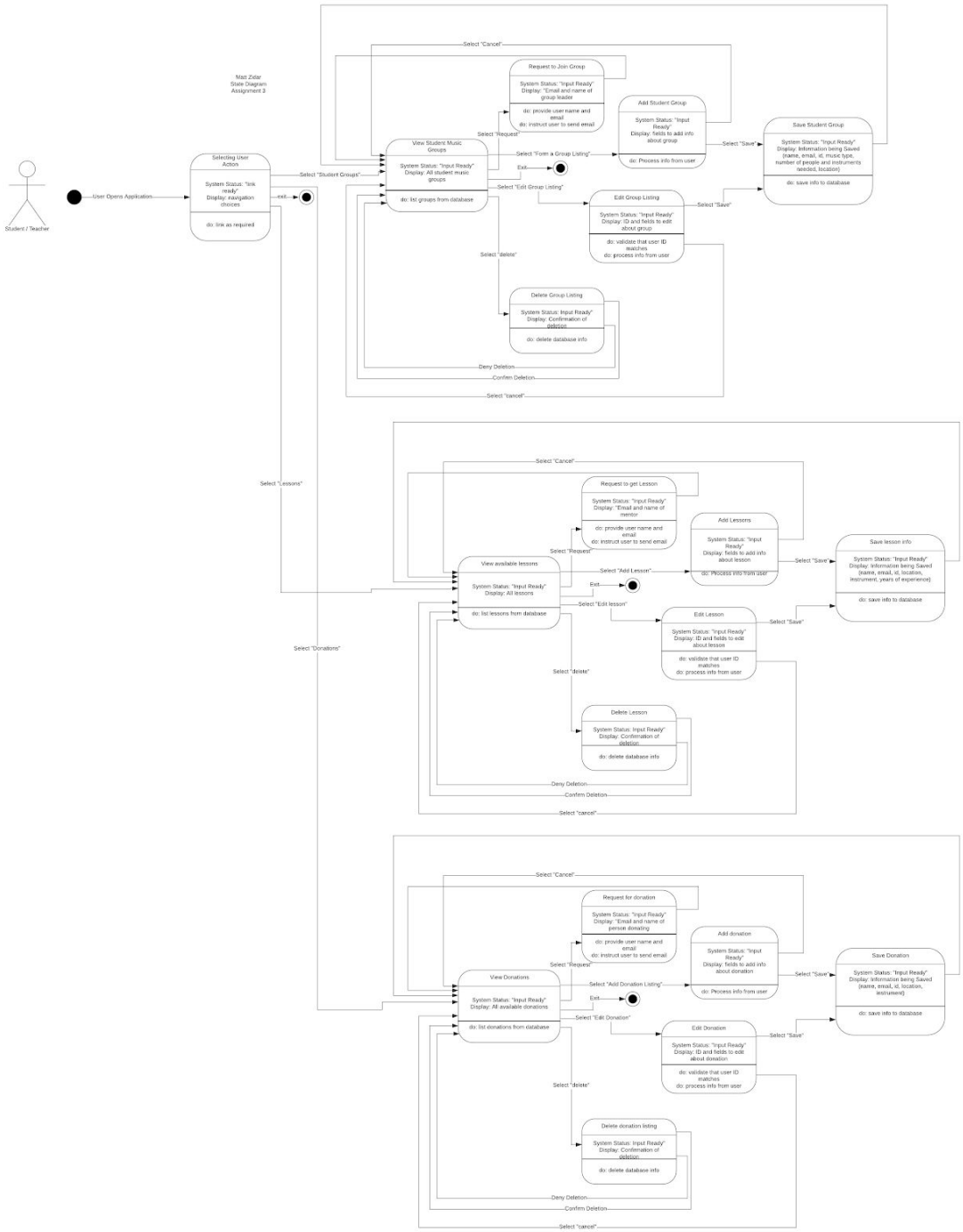
- Search that the student does does not return any results - the system prompts the user to enter different criteria for a different search.

2. Design Class Diagram - PDF included in submission and on Github as well



3. System Sequence Diagrams - PDF files included in submission and on Github as well, but not in this document

4. Statechart - PDF file included in submission and on Github as well.



5. Mockup UI and Write-Ups - Diagrams only included in PDF form and will not be included in this document. However, write-up for this section is below.

The mock-ups show affordance and visibility because each task is split up into a different button. As soon as a button is pushed, the user is prompted to enter information and are guided through the process, or a new page opens with self-explanatory instructions as to what the user should do next. For example, when adding a listing of any type, the boxes display what kind of information should be included in each text box, and the system specifies whether it wants the user to write it as a string, or if they have to select from a list of options. There are not too many buttons per page and the layout is simple, so the user gets a good idea of what exactly is the next step after each action and what each page can do.

The layout for many of the different pages is consistent. Button and table placement remain relatively the same across pages of similar function. This makes it easy for the user to understand how the whole application works. The system also gives the user information when necessary if it is not already included on the screen. For example, when a listing is edited, created, or deleted, the system tells them that this has happened successfully and they can immediately see the result in the table. Error handling will work well. The system will inform the user exactly what is wrong with the information they input, and will not require them to re-enter everything they just filled out if they need to go back and redo something. The 'Cancel' button allows for easy reversal of actions. If a user decides to cancel a listing that they are creating or editing or deleting, they can easily do so by hitting the cancel button. The user will be informed that the action has been cancelled successfully. The user should feel a sense of control when they use the application. For their listings, they can easily edit a lot of the information or do any necessary actions like deletion. All that is required is that the user remembers their ID. The program easily reduces the short term memory load. For example, if a user is trying to edit their information in a listing, the screen will display what their information currently is so they don't have to remember everything. A consistent layout also helps for this as the user does not have to remember as much about the application. Shortcuts can be used easily for users if they are completing a certain action. For example, if they are trying to request a donation, they can easily then move to another page by clicking the buttons up top instead of closing out the page they are currently in then then moving to the other page.

6.

This program is modular because each of the different operations that are required for lessons, donations, and forming music groups such as adding a listing, editing or deleting a listing, or requesting a certain listing are broken up into different functions. Each of the variables inside of the class are all private and can only be accessed through getter methods in specific operations. Additionally, since many of these operations are fairly similar such as adding a listing for donations and for lessons, these functions can possibly be reused as I am developing the rest of the program, however with a couple parameters changed.

The main data structure that is being used is the PostgreSQL database, with all information being stored and retrieved from there. The reason that I am using this in my application is that many people in a specific area will have the ability to use the application, and operations such as creating a listing on one user's end needs to be able to show up whenever someone else uses the

application. The use of the PostgreSQL database will allow for any changes being made to listings to appear on the application for every user who tries to use it, despite all being in different remote locations.

While I have yet to implement it, I am planning on using a sorting algorithm such as Quicksort to sort the listings based on what the user is trying to search for. This will allow for the user to search for listings based on certain criteria such as instrument name, and quickly get a full list of the results. Part of using a database is making sure that the application makes good queries, and sorting and searching this way will allow for good queries to be made.

7.

I do not plan on using any debugging or testing tools. The main area of concern with my program is making sure that when users are entering data into the system, it must be accurate and completed fully. This is important because some of these fields will be used for sorting and searching and if data is not entered accurately, it might not provide the user with the correct output based on what they are looking for. I am planning on making a lot of the fields where users can enter information such as instrument name or years of experience a box where they can select from a list of items (in this example, select from a list of instruments such as trumpet, trombone, oboe, or select from a list of 1-20 for years of experience). This will make sure that some fields are inputted correctly and without spelling or datatype errors.

Note: All of the functions that are listed in this table apply for every different major function of the program, such as with donation listings, lesson listings, and student music group listings. These are methods that the user can do to try and break the application, and what type of error message they will get from this. Major functions are included twice as one shows what happens when an error is raised, and one shows what will happen when there is no error.

Functionality Tested	Inputs	Expected Output	Actual Output
Add Listing	None - ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘ Or Some - ‘John Smith’, ‘Trumpet’, ‘ ‘, ‘ ‘	ERROR: Please enter information in all fields before submission!	N/A
Add Listing	Used valid values for all fields	Listing for donation, music group, or lesson entered successfully	N/A
Validate ID before editing or deleting	Entered ID matches ID stored in listing	User successfully edits or deletes their	N/A

listing		listing	
Validate ID before editing or deleting listing	Non-Matching ID - User enters '1234' when ID is '2242'	ERROR: Please enter valid user ID to edit or delete this listing. You have x attempts remaining!	N/A
Search for Listings	User makes search based on mentor having 10 years of experience: yearsOfExperience = '10' but search is not returned	ERROR: your search has returned 0 results. Please refine your search criteria before searching again!	N/A
Search for Listings	User makes search based on criteria that exist in table	List of results are returned accurately	N/A

8. Prototype - Code so far is located in Github and prototype is on VM. Link to repo and project on VM is listed at the top of this document