TEAM NOVA ELITE

CSCE 4901 – Computer Science Capstone

Design Document 3/11/2016

Table of Contents:

Introduction

This document is a Design Document deliverable created to compile the design diagrams to begin planning how the team is going to physically build the application through code. Team Nova Elite is the team behind the construction of the proposed project and will also be tasked with refining the base product and any extra details that arise throughout development.

The Team consists of the following members who each currently attend the University of North Texas in pursuit of a degree in Computer Science: Jason Hoang, Imran Akhtar, Kai-Chuan Chan, and Sabrin Thamed. We will use this project as an opportunity to learn more knowledge and skills about our science along with how to produce an application that is useful in the real world. Team Nova Elite is dedicated to ensure that the final base product is fully functional within the time given to complete the project.

The rest of the requirements specification document will serve as a tentative list of the main requirements for the project. It will go into further detail about each requirement, how they affect the field of education, and address the issues involved for producing each requirement. Each requirement is also subject to change throughout development based on any issues encountered or change in ideas or need by the client.

Description of the Project:

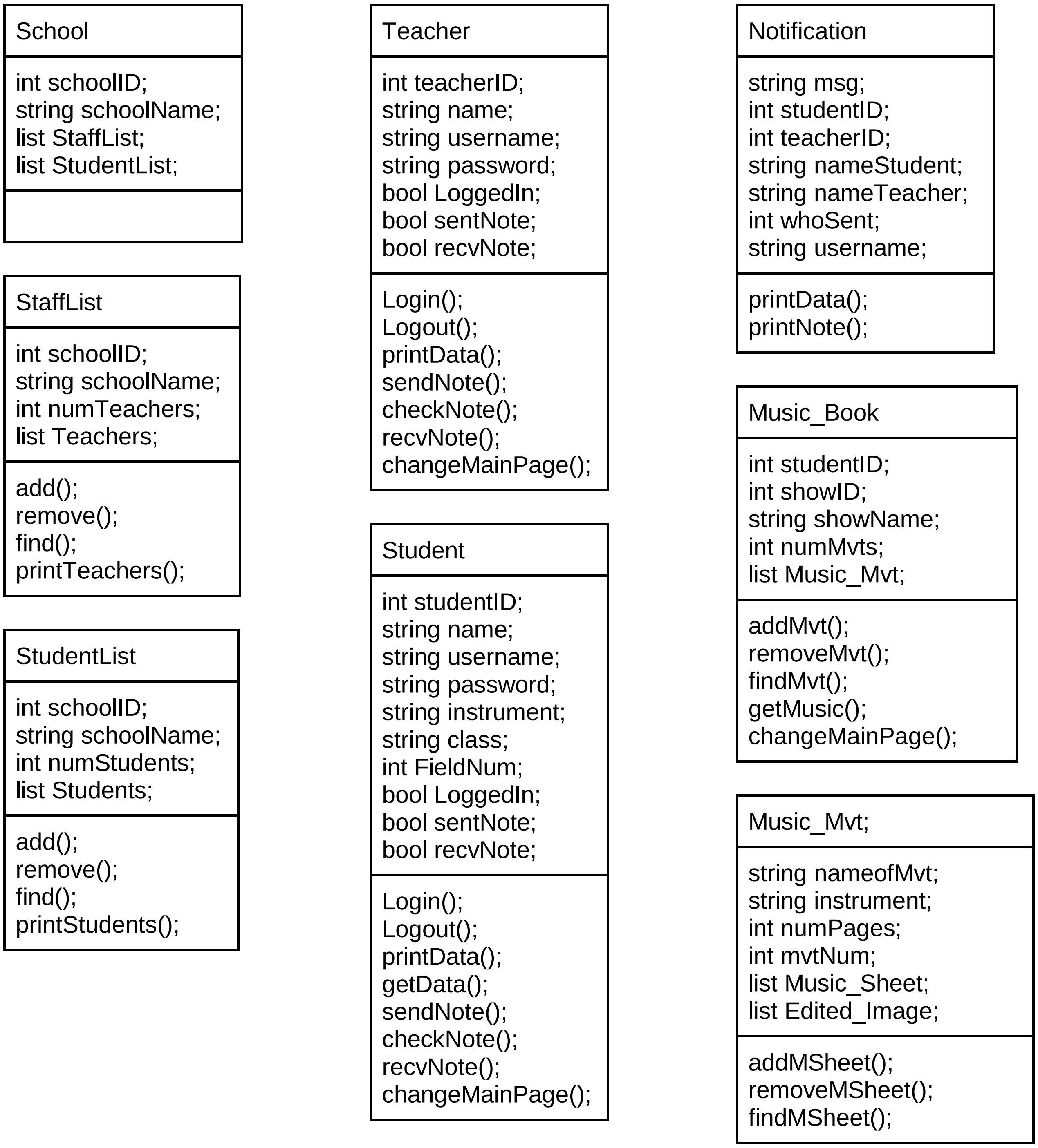
This project consists of a teacher app and student apps that communicate directly with each other, giving the teacher the ability to guide the students who spread across the marching field/rehearsal room via a single device and allowing students to ask questions and take notes with ease. The program will focus on compiling all of the paperwork, sheet music, and coordinate sheets involved with students learning, rehearsing, and performing a marching band show onto their own personal devices and tablets. This will make the process of teaching easier and help the students stay organized with all of their papers. All apps will also have access to vital learning tools for a performing ensemble so that the students can practice with the app on their own. These apps will be designed for marching bands across the United States, but they can be adjusted to satisfy other indoor rehearsals for any type of ensemble.

The Purpose of the Project:

Team Nova Elite is tasked with producing an application that serves a portion of the general public. The students who were assembled for this team are working together for the first time in their careers and therefore have put together a list of objectives for the team to achieve alongside producing the main application:

- Build an application that serves music educators in the field of marching band at a high and efficient level.
- Form an inviting and hard working environment for each member of the group to feel comfortable working with and complete a successful project.
- Produce and understand the other side to producing an application of this scale in terms of documentation and compiling all of the other team members' work and data.
- Strive for punctual completion of each requirement and feature of the project with the initial focus on functionality.
- Learn and be comfortable with new IDEs and programming languages that are commonly used in the field.

Class Diagram (Classes)



```
School
int schoolID;
string schoolName;
list StaffList;
list StudentList;
```

```
StaffList
int schoolID;
string schoolName;
int numTeachers;
list Teachers;

add();
remove();
find();
printTeachers();
```

```
StudentList
int schoolID;
string schoolName;
int numStudents;
list Students;

add();
remove();
find();
printStudents();
```

```
Teacher
int teacherID;
string name;
string username;
string password;
bool LoggedIn;
bool sentNote;
bool recvNote;

Login();
Logout();
printData();
sendNote();
checkNote();
recvNote();
changeMainPage();
```

```
Student
int studentID;
string name;
string username;
string password;
string instrument;
string class;
int FieldNum;
bool LoggedIn;
bool sentNote;
bool recvNote;

Login();
Logout();
printData();
getData();
sendNote();
checkNote();
recvNote();
changeMainPage();
```

```
Notification
string msg;
int studentID;
int teacherID;
string nameStudent;
string nameTeacher;
int whoSent;
string username;

printData();
printNote();
```

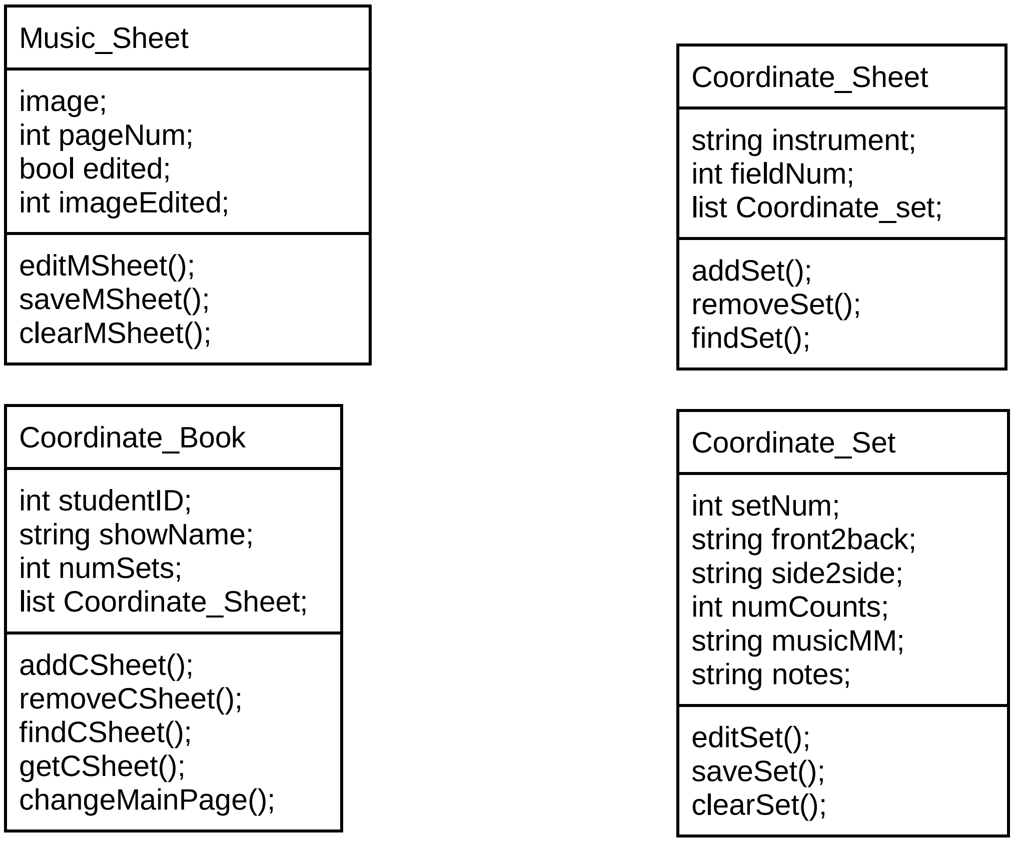
```
Music_Book
int studentID;
int showID;
string showName;
int numMvts;
list Music_Mvt;

addMvt();
removeMvt();
findMvt();
getMusic();
changeMainPage();
```

```
Music_Mvt;
string nameofMvt;
string instrument;
int numPages;
int mvtNum;
list Music_Sheet;
list Edited_Image;

addMSheet();
removeMSheet();
findMSheet();
```

Class Diagram (Classes continued):



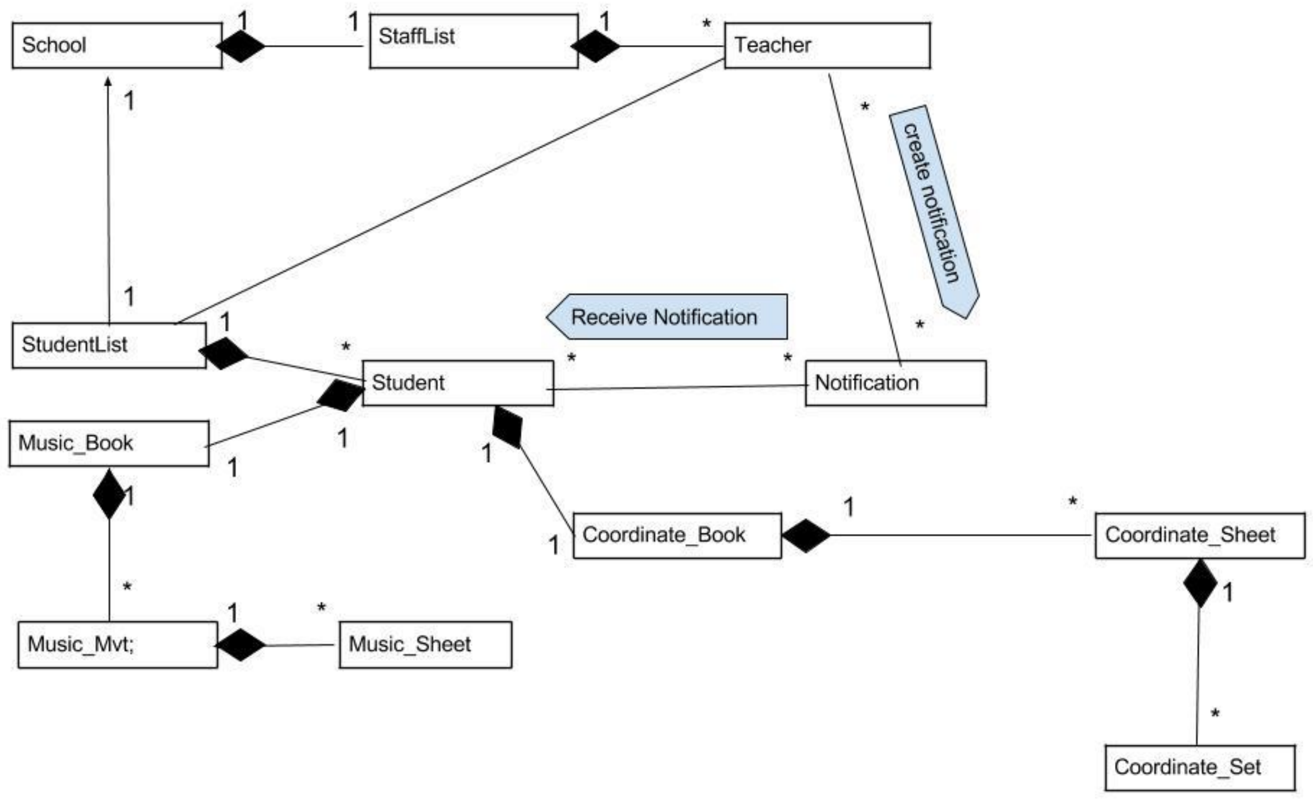
```
Music_Sheet
---
image;
int pageNum;
bool edited;
int imageEdited;
---
editMSheet();
saveMSheet();
clearMSheet();
```

```
Coordinate_Sheet
---
string instrument;
int fieldNum;
list Coordinate_set;
---
addSet();
removeSet();
findSet();
```

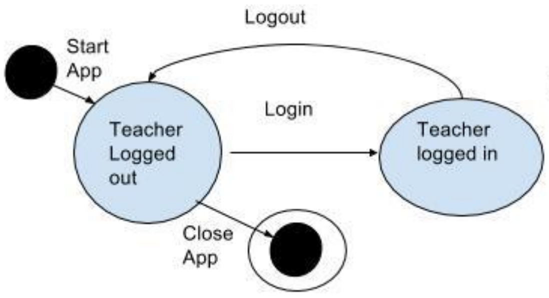
```
Coordinate_Book
---
int studentID;
string showName;
int numSets;
list Coordinate_Sheet;
---
addCSheet();
removeCSheet();
findCSheet();
getCSheet();
changeMainPage();
```

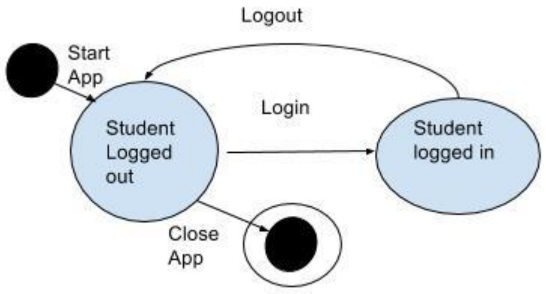
```
Coordinate_Set
---
int setNum;
string front2back;
string side2side;
int numCounts;
string musicMM;
string notes;
---
editSet();
saveSet();
clearSet();
```

Class Diagram(Class Relationships):

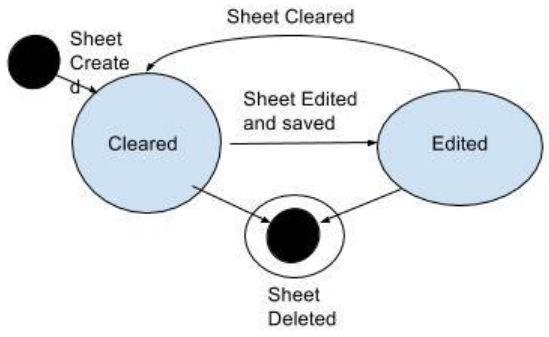


State Diagrams:

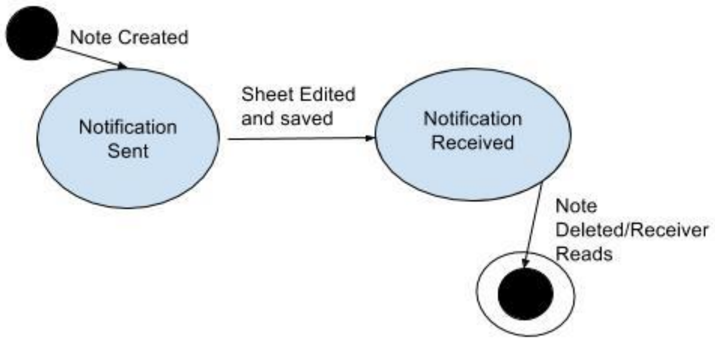




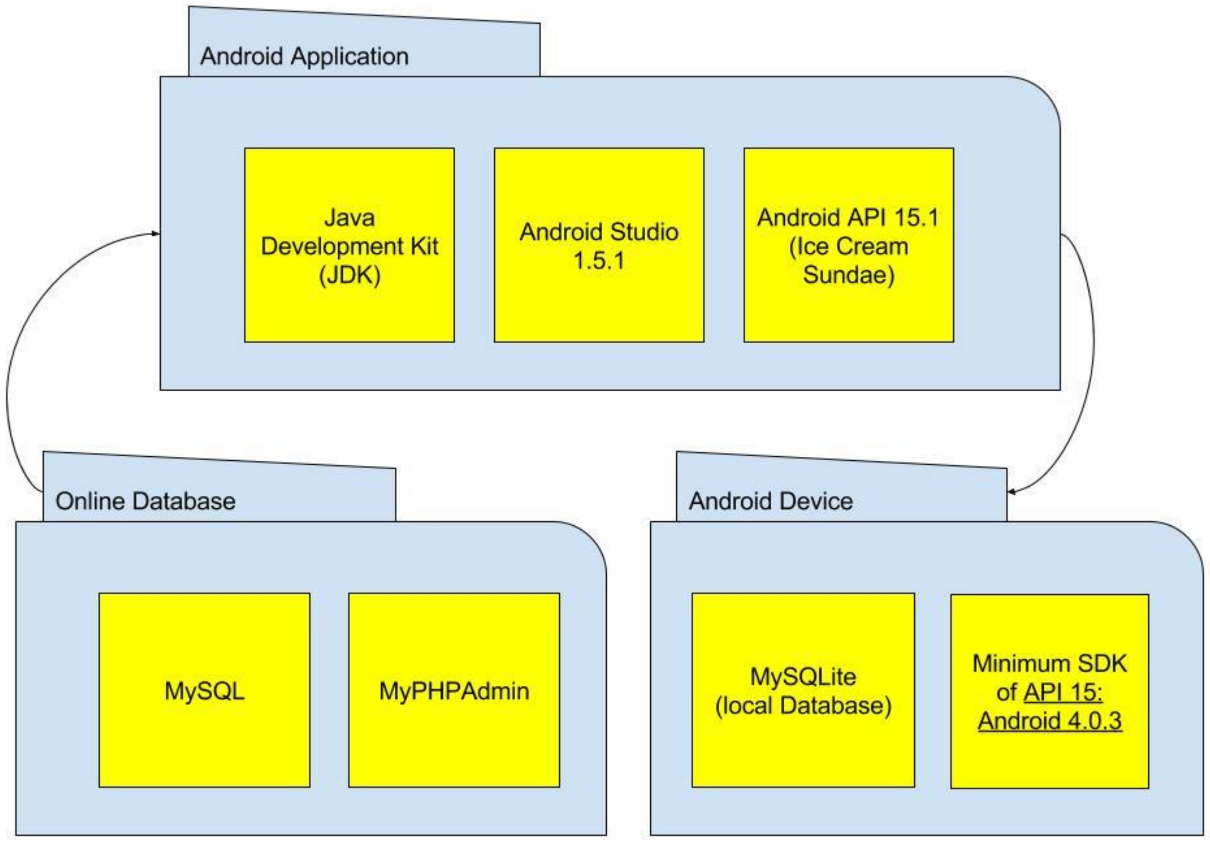
Music Sheet/Coordinate Sheet



Notification States



Package Diagram:


Android Application
Java Development Kit (JDK)
Android Studio 1.5.1
Android API 15.1 (Ice Cream Sundae)
Online Database
MySQL
MyPHPAdmin
Android Device
MySQLite (local Database)
Minimum SDK of API 15: Android 4.0.3

Pseudocode(for the class functions):

```
add(){              //general add function for all possible items for the database
        get input from user according to what data is needed for the item being added;
        store input into temporary variables used to search;
        int tempInt;
        string tempStr;
        bool tempBool;
        find(); //if item is already in the database
        if(itemFound)
                //do not add to list
                print("item already in database");
        else
                create node with input about the new item;
                add node to list of corresponding items;
                list.numItems++;
        return;
}

remove(){           //general remove function for all possible items for the database
        get input from user according to which item(s) is/are being deleted;
        store input into temporary values used to search;
        int tempInt;
        string tempStr;
        bool tempBool;
        find(); //searches for the item(s) in question
        if(itemFound)
                clear the memory used by the node;
                safely delete the node;
                list.numItems--;
        else
                //do not delete anything
                printf("Item not found in database");
        return;
}

find(){             //general find function for all possible items for the database
        //searches the specified database or list for the item(s) in question and returns where it is found, or NULL
        gets input from user about item being searched for and which database it could be in;
        selects the appropriate database to search and stores it to a temporary list;
        list tempList;
        for(i=0; i<numItems; i++)
                //traverse the list until the item is found or not
                if(user search input == list.item properties)
                        //item found
                        return item location;
                else
                        //continue search
        if(i == numItems)
                //item was not found
                return NULL;
}
```

```
printList(){        //general printList function for all possible items for the database
    get input from user according to what data is needed for the list being printed;
    list tempList;
    for(i=0; i<numItems; i++)
        //traverse list and print every item with the requested details
        print(list.Item);
}

printItem(){        //general printItem function for all possible items for the database
    get input from user according to what data is needed for the item being printed;
    item tempItem;
    tempItem = find(item);
    print(item.ItemDetails);
}

Login(){
    get username and password input from user;
    validate the credentials with the online database;
    if(validated)
        //user is logged in
        user.loggedIn = TRUE;
        application switches activity screens to the main page;
    else
        print("ERROR: username or password incorrect");
}

Logout(){
    get username and password from user;
    validate the credentials with the online database;
    if(validated)
        //user is logged in
        user.loggedIn = FALSE;
        application switches activity screens to the Login page;
    else
        //this should never happen
        print("ERROR: unable to logout");
}

getData(){          //general getData function for all possible items for the database
    //this function is similar to print by using find, but instead of printing data it will return specific data
    //to another function that will use it
    user inputs the item or list that they want data from;
    find(item/list);
    return item.data;
}
```

```
sendNote(){
        user types message and/or selects message details;
        getData(user);
        getData(receiver);
        stores above input and data into note;
        addNote(user.Data, receiverData, message);
        set notification flags for the receiver;
        send to receiver via online database;
}

checkNote(){
        //since we want notifications to appear in real-time, this function will be called often throughout the
program
        check if note is in user inbox;
        if(noteFound)
                set large visual notification flags on screen;
                delay(3 seconds);
                shrink visual notification flag to appropriate place on the app menu bar;
        else
                //continue
}

recvNote(){
        //this is called when user accesses the notifications option
        pull notification from database;
        print(note.SenderData);
        print(message);
        give receiver option to reply;
        if(wantToReply)
                sendNote;
        else
                close notification;
}
```

```
editMSheet(){      //MSheet interchangable with Set(Coordinate Set)
    //this function uses android's SQLite to store data to the user's device
    check device database if a copy of MSheet exists;
    if(MSheet)
        pull MSheet from device database;
        editing = TRUE;
    else
        create copy of MSheet;
        user inputs any edits using the application touch screen features;
        editing = TRUE;
    while(editing)
        user is given option to save, clear, quit, or continue editing from menu bar;
        if(save)
            saveMSheet(copy of MSheet);
        else if(clear)
            clearMSheet(copy of MSheet);
        else if(quit)          //quit could mean logging out or changing activity pages
            give user second chance option to save before quiting;
            if(save)
                saveMSheet(copy of MSheet);
                editing = FALSE;
            else if(cancelQuit)
                editing = TRUE;
            else
                //quit without saving
                editing = FALSE;
        else
            editing = TRUE;

    return toTargetLocation;
}

saveMSheet(copy of MSheet){        //MSheet interchangable with Set(Coordinate Set)
    //this function uses android's SQLite to store data to the user's device
    gets copy of MSheet from function call;
    SQLite call to save copy to device database;
    update device database to indicate that a copy exists;
}

clearMSheet(copy of MSheet){       //MSheet interchangable with Set(Coordinate Set)
    //this function uses android's SQLite to store data to the user's device
    gets copy of MSheet from function call;
    clears memory used by copy of MSheet;
    delete copy of MSheet from device database;
    update device database to indicate that a copy does not exist;
}

ChangeMainPage(){
    user inputs target location request;
    //this involves a touch screen swipe to either the left or right depending on which page the user is on
    android functions switches activity pages to target location;
    updates user data to save user location within app;
}
```

```
//Activity pages that utilize the above funtions the most
LoginPage(){
        user chooses to login to the system or exit the application;
        switch(choice){
                case Login:
                        Login();
                case Exit:
                        exit();
        }
}

MainPage1(){    //for the music page
        user chooses actions within application;
        switch(choice){
                case ChangeMainPage:
                        ChangeMainPage();
                case menuBar:
                        MenuBar();
                case Logout:
                        Logout();
                case Exit:
                        exit();
        }
}

MainPage2(){    //for the coordinates
        user chooses actions within application;
        switch(choice){
                case ChangeMainPage:
                        ChangeMainPage();
                case menuBar:
                        MenuBar();
                case Logout:
                        Logout();
                case Exit:
                        exit();
        }
}

MusicMenu(){
        user selects action;
        switch(choice)
        {
                case selectItem:
                        user selects the music item that they wish to view;
                        if(find()) //finds the music book in the database
                        {
                                getData();          //retrieves the item
                                printItem();        //displays to the screen
                        }
                        else
                        {
                                "ERROR: item not found";
                }
            case addItem:
                add();                  //user can add new music
            case removeItem:
                remove();               //or remove music
            case edit:
                editMSheet();
            case clear:
                clearMSheet();
            case save:
                saveMSheet();
            case quit:
                exit();
        }
}

CoordinateMenu(){
        user selects action;
        switch(choice)
        {
            case selectItem:
                user selects the coordinate item that they wish to view;
                if(find()) //finds the coordinate sheet in the database
                {
                    getData();          //retrieves the item
                    printItem();        //displays to the screen
                }
                else
                {
                    "ERROR: item not found";
                }
            case addItem:
                add();                  //user can add new coordinate sheets/sets
            case removeItem:
                remove();               //or remove coordinate sheets/sets
            case edit:
                editCSet();
            case clear:
                clearCSet();
            case save:
                saveCSet();
            case quit:
                exit();
        }
}
```

```
StudentMenuBar(){
        user chooses which function they want from the menu bar;
        PrintList();        //list of choices for the menu
        switch(menuChoice){
                case Notification: //for the student, their notifcation only goes to the teacher, so they do not
have to select where it goes
                        getData();          //for the user
                        if(checkNote())     //if there is a note
                        {
                                recvNote();
                        }
                        else if(user wants to send note)
                        {
                                getData();          //for the receiver
                                sendNote();
                        }
                case MusicItem:  //user selects an item that they want to view or edit
                        if(user is on the Music main page)
                        {
                                MusicMenu();
                        }
                        else if(user is on the Music main page)
                        {
                                CoordinateMenu();
                        }
                case musicTool:
                        metronome();
                        tuner();
                case quit:
                        exit();
        }
}

TeacherMenuBar(){
        user chooses which function they want from the menu bar;
        PrintList();        //list of choices for the menu
        switch(menuChoice){
                case MusicItem:  //user selects an item that they want to view or edit
                        if(user is on the Music main page)
                        {
                                MusicMenu();
                        }
                        else if(user is on the Music main page)
                        {
                                CoordinateMenu();
                        }
                case musicTool:
                        metronome();
                        tuner();
                case Attendance:
                        Attendance();
                case quit:
                        exit();
        }
}

Attendance(){
        getData();          //for all the students and teachers within the school's database
        printList();        //for all users
        do{
                visually marks everyone currently logged in(green) or not(red);
                teacher selcts an action;
                switch(choice)
                {
                        case view:          //selects a student
                                teacher can select students to view information or send notifications to;
                                getData(); for the student selected
                        case addAccount:
                                add();              //the teacher can add
                        case removeAccount:
                                remove();           //or remove students/staff from the roster here
                        case notification:
                                if(checkNote())     //if there is a note
                                {
                                        recvNote();
                                }
                                else if(user wants to send note)
                                {
                                        getData();          //for the receiver
                                        sendNote();
                                }
                }
        }while(choice != QUIT)
}
```