Criterion C: Development

MainActivity class

```
public class MainActivity extends AppCompatActivity {
```

This class extends AppCompatActivity due to all Activities having to have a default constructor.

```
//Navigation Menu
{
    DrawerLayout drawerLayout = (DrawerLayout) findViewById(R.id.drawerLayout);

    findViewById(R.id.imageMenu).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { drawerLayout.openDrawer(GravityCompat.START); }
    });

    NavigationView navigationView = (NavigationView) findViewById(R.id.navigationView);
    navigationView.setItemIconTintList(null);

    NavController navController = (NavController) Navigation.findNavController( activity: this,
    NavigationUI.setupWithNavController(navigationView, navController);

    TextView textTitle = (TextView) findViewById(R.id.textViewTitle);

    navController.addOnDestinationChangedListener(new NavController.OnDestinationChangedList
        @Override
        public void onDestinationChanged(@NonNull NavController controller, @NonNull NavDest
            textTitle.setText(destination.getLabel());
        }
    });
}
```

This is the code for the navigation menu in the app. It Creates a drawerLayout object so that it can be interacted with. The set on click listener is used for opining the drawerLayout when the open menu button is clicked. The middle area of the code is to create and load the menu. The destination change listener is to change the fragment to the one that is clicked on the menu.

```
//Speed Dial
{
    SpeedDialView speedDialView = (SpeedDialView) findViewById(R.id.speedDialView);

    speedDialView.addActionItem(
            new SpeedDialActionItem.Builder(R.id.addAssignmentFAB, R.drawable.ic_baseline_post_add_24)
                    .setFabBackgroundColor(getColor(R.color.Primary))
                    .setLabel("Add New Assignment")
                    .setLabelClickable(false)
                    .create()
    );

    speedDialView.addActionItem(
            new SpeedDialActionItem.Builder(R.id.addClassFAB, R.drawable.ic_baseline_add_to_photos_24)
                    .setFabBackgroundColor(getColor(R.color.Primary))
                    .setLabel("Add New Class")
                    .setLabelClickable(false)
                    .create()
    );

    speedDialView.setOnActionSelectedListener(new SpeedDialView.OnActionSelectedListener() {
        @Override
        public boolean onActionSelected(SpeedDialActionItem speedDialActionItem) {
            if (speedDialActionItem.getId() == R.id.addClassFAB) {
                startActivity(new Intent( packageContext: MainActivity.this, addClass.class));
                return false;
            }
            else if (speedDialActionItem.getId() == R.id.addAssignmentFAB) {
                startActivity(new Intent( packageContext: MainActivity.this, addAssignment.class));
                return false;
            }
            return false;
        }
    });
}
```

The speed dial is part of the leinardi floating action button speed dial library (https://github.com/leinardi/FloatingActionButtonSpeedDial). This library is used as it is a simple way of adding an effective speed dial that gets the desired results while simplifying the required implementation. This makes the speed dial less problem prone as this is a well-used library for speed dial implementation.

TimeTableFragment class

```
public class TimeTableFragment extends Fragment {
```

This class extends Fragment due to all Fragments having to have a default constructor.

```
public static final int TOTALDAYS = 7 ;

String dateSetString;
Integer dateSetInt;
TextView dateClasses;
Button ButtonLeft, ButtonRight;
```

Defines all variables used in this class as well as all the objects that need to be interacted with. Use of public static variable for TOTALDAYS as so that other classes can change this and view it if a feature comes out for needing the changes the number of timetable days.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_time_table, container,  attachToRoot: false);
```

onCreateView() is needed for a fragment to display a XML sheet as well as all interactions with this fragment take place in this method.

```
dateClasses = view.findViewById(R.id.textViewDateOfClass);
ButtonLeft = view.findViewById(R.id.buttonLeft);
ButtonRight = view.findViewById(R.id.buttonRight);
```

This sets all objects that will be used in the code.

```
//Back in date button click
ButtonLeft.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        dateSetString = dateClasses.getText().toString();
        dateSetInt = Integer.parseInt(dateSetString);

        if (dateSetInt != 1)
            dateSetInt--;
        else dateSetInt = TOTALDAYS;

        dateSetString = dateSetInt.toString();
        dateClasses.setText(dateSetString);
    }
});
```

This setOnClickListener() allows you to cycle through the timetable days changing what day is being viewed in the list of timetable classes.

```
//Forward in date button click
ButtonRight.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        dateSetString = dateClasses.getText().toString();
        dateSetInt = Integer.parseInt(dateSetString);

        if (dateSetInt != TOTALDAYS)
            dateSetInt++;
        else dateSetInt = 1;

        dateSetString = dateSetInt.toString();
        dateClasses.setText(dateSetString);
    }
});
```

This is the forward version of the back in date setOnClickListener().

ClassesFragment class

```
public class ClassesFragment extends Fragment {
```

Extends Fragment class.

```
ListView listViewClasses;
DBClassTable dataBase;
ArrayAdapter<UserClass> userClassArrayAdapter;
View view;
```

Defining all objects that are used in the class.

```
@Override
public void onResume() {
    super.onResume();
    refreshList();
}
```

When the fragment resumes being in focus it refreshes list of class shown in the userClassArrayAdapter. This is used so when you add a class to the list it added it to the shown list with out having to close the fragment.

```
public void refreshList() {
    userClassArrayAdapter = new ArrayAdapter<UserClass>(getActivity().getApplicationContext(),
    listViewClasses.setAdapter(userClassArrayAdapter);
}
```

refreshList() method gets all classes in the classes database and appends them to the ArrayAdapter so that all the classes in the database are shown in the fragment.

UserClass Class

```
public UserClass(int id,
                 String nameOfClass,
                 String nameOfTeacher,
                 String teacherEmail,
                 int day1,
                 int day2,
                 int day3,
                 int day4,
                 int day5,
                 int day6,
                 int day7,
                 String day1Time,
                 String day2Time,
                 String day3Time,
                 String day4Time,
                 String day5Time,
                 String day6Time,
                 String day7Time) {
    this.id = id;
    this.nameOfClass = nameOfClass;
    this.nameOfTeacher = nameOfTeacher;
    this.teacherEmail = teacherEmail;
    this.day1 = day1;
    this.day2 = day2;
    this.day3 = day3;
    this.day4 = day4;
    this.day5 = day5;
    this.day6 = day6;
    this.day7 = day7;
    this.day1Time = day1Time;
    this.day2Time = day2Time;
    this.day3Time = day3Time;
    this.day4Time = day4Time;
    this.day5Time = day5Time;
    this.day6Time = day6Time;
    this.day7Time = day7Time;
}
```

This class is very simple with just a constructor for setting the variables and a toString() method to make it presentable in the class and timetable fragment.

```java
@Override
public String toString() {
    return "UserClass{" +
            "id=" + id +
            ", nameOfClass='" + nameOfClass + '\'' +
            ", nameOfTeacher='" + nameOfTeacher + '\'' +
            ", teacherEmail='" + teacherEmail + '\'' +
            ", day1=" + day1 +
            ", day2=" + day2 +
            ", day3=" + day3 +
            ", day4=" + day4 +
            ", day5=" + day5 +
            ", day6=" + day6 +
            ", day7=" + day7 +
            ", day1Time='" + day1Time + '\'' +
            ", day2Time='" + day2Time + '\'' +
            ", day3Time='" + day3Time + '\'' +
            ", day4Time='" + day4Time + '\'' +
            ", day5Time='" + day5Time + '\'' +
            ", day6Time='" + day6Time + '\'' +
            ", day7Time='" + day7Time + '\'' +
            '}';
}
```

UserAssignment class

```java
public UserAssignment(int id, String assignmentName, String assignmentDescription, String assignme
    this.id = id;
    this.assignmentName = assignmentName;
    this.assignmentDescription = assignmentDescription;
    this.assignmentDate = assignmentDate;
    this.assignmentTime = assignmentTime;
}
```

This class is also very simple with just a constructor to make the class and a toString method to make it presentable.

```java
@Override
public String toString() {
    return "UserAssignment{" +
            "id=" + id +
            ", assignmentName='" + assignmentName + '\'' +
            ", assignmentDescription='" + assignmentDescription + '\'' +
            ", assignmentDate='" + assignmentDate + '\'' +
            ", assignmentTime='" + assignmentTime + '\'' +
            '}';
}
```

DBAssignmentTable Class

```
//Creates TABLE for assignments, This is currently only supporting NAME of a
@Override
public void onCreate(SQLiteDatabase db) {
    String createTableStatement = "CREATE TABLE " + ASSIGNMENT_TABLE + " ("
            COLUMN_ASSIGNMENT_NAME + " TEXT, " +
            COLUMN_ASSIGNMENT_DESCRIPTION + " TEXT, " +
            COLUMN_ASSIGNMENT_DATE + " TEXT, " +
            COLUMN_ASSIGNMENT_TIME + " TEXT)";

    db.execSQL(createTableStatement);
}
```

This method is used when the app is first loaded and so it creates a new database using db.execSQL.

```
public boolean addUserAssignment(UserAssignment userAssignment){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();

    // input userAssignment fields into there respective columns
    cv.put(COLUMN_ASSIGNMENT_NAME, userAssignment.assignmentName);
    cv.put(COLUMN_ASSIGNMENT_DESCRIPTION, userAssignment.assignmentDescriptior
    cv.put(COLUMN_ASSIGNMENT_DATE, userAssignment.assignmentDate);
    cv.put(COLUMN_ASSIGNMENT_TIME, userAssignment.assignmentTime);

    // inset into table.
    long insert = db.insert(ASSIGNMENT_TABLE,  nullColumnHack: null, cv);

    return insert != -1;
}
```

This method inputs the variables stored in the UserAssignment that was inputted into the method. It dose this by using the ContentValues class to interface with the database.

```
public boolean deleteUserAssignment(Integer ID){
    SQLiteDatabase database = this.getWritableDatabase();
    String queryString = "DELETE FROM " + ASSIGNMENT_TABLE + " WHERE " + COLUMN_ID + " = " + ID;
    Cursor cursor = database.rawQuery(queryString,  selectionArgs: null);

    return !cursor.moveToFirst();
}
```

This method deletes a row on the database by using a cursor to query the database for a specific ID then deleting that row.

```
public List<UserAssignment> getAllAssignments(){
    List<UserAssignment> returnList = new ArrayList<>();
    String queryString = "SELECT * FROM " + ASSIGNMENT_TABLE;
    SQLiteDatabase database = this.getReadableDatabase();
    Cursor cursor = database.rawQuery(queryString,  selectionArgs: null);

    if (cursor.moveToFirst()) {

        do{
            int assignmentID = cursor.getInt( columnIndex: 0);
            String assignmentName = cursor.getString( columnIndex: 1);
            String assignmentDescription = cursor.getString( columnIndex: 2);
            String assignmentDate = cursor.getString( columnIndex: 3);
            String assignmentTime = cursor.getString( columnIndex: 4);

            UserAssignment userAssignment = new UserAssignment(assignmentID, assignmen
            returnList.add(userAssignment);
        }while (cursor.moveToNext());
    }

    //Closes Database and cursor as to not take up usage of database.
    database.close();
    cursor.close();

    return returnList;
}
```

This method iterated through the database getting each row and storing that in a list of UserAssignments then returning that list.

DBClassTable class

```
@Override
public void onCreate(SQLiteDatabase db) {
    String createTableStatement = "CREATE TABLE " + CLASS_TABLE + " (" +
            COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
            COLUMN_CLASS_NAME + " TEXT, " +
            COLUMN_TEACHER_NAME + " TEXT, " +
            COLUMN_TEACHER_EMAIL + " TEXT, " +
            COLUMN_DAY_1 + " INTEGER, " +
            COLUMN_DAY_2 + " INTEGER, " +
            COLUMN_DAY_3 + " INTEGER, " +
            COLUMN_DAY_4 + " INTEGER, " +
            COLUMN_DAY_5 + " INTEGER, " +
            COLUMN_DAY_6 + " INTEGER, " +
            COLUMN_DAY_7 + " INTEGER, " +
            COLUMN_DAYTIME_1 + " TEXT, " +
            COLUMN_DAYTIME_2 + " TEXT, " +
            COLUMN_DAYTIME_3 + " TEXT, " +
            COLUMN_DAYTIME_4 + " TEXT, " +
            COLUMN_DAYTIME_5 + " TEXT, " +
            COLUMN_DAYTIME_6 + " TEXT, " +
            COLUMN_DAYTIME_7 + " TEXT)";

    db.execSQL(createTableStatement);
}
```

This method is used when the app is first loaded and so it creates a new database using db.execSQL.

```java
public boolean addUserClass(UserClass userClass){

    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();

    cv.put(COLUMN_CLASS_NAME, userClass.nameOfClass);
    cv.put(COLUMN_TEACHER_NAME, userClass.nameOfTeacher);
    cv.put(COLUMN_TEACHER_EMAIL, userClass.teacherEmail);
    cv.put(COLUMN_DAY_1, userClass.day1);
    cv.put(COLUMN_DAY_2, userClass.day2);
    cv.put(COLUMN_DAY_3, userClass.day3);
    cv.put(COLUMN_DAY_4, userClass.day4);
    cv.put(COLUMN_DAY_5, userClass.day5);
    cv.put(COLUMN_DAY_6, userClass.day6);
    cv.put(COLUMN_DAY_7, userClass.day7);
    cv.put(COLUMN_DAYTIME_1, userClass.day1Time);
    cv.put(COLUMN_DAYTIME_2, userClass.day2Time);
    cv.put(COLUMN_DAYTIME_3, userClass.day3Time);
    cv.put(COLUMN_DAYTIME_4, userClass.day4Time);
    cv.put(COLUMN_DAYTIME_5, userClass.day5Time);
    cv.put(COLUMN_DAYTIME_6, userClass.day6Time);
    cv.put(COLUMN_DAYTIME_7, userClass.day7Time);

    long insert = db.insert(CLASS_TABLE,  nullColumnHack: null, cv);

    return insert != -1;
}
```

This method inputs the variables stored in the UserClass that was inputted into the method. It dose this by using the ContentValues class to interface with the database.

```
public List<UserClass> getAllClasses(){
    List<UserClass> returnList = new ArrayList<>();
    String queryString = "SELECT * FROM " + CLASS_TABLE;
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(queryString,  selectionArgs: null);

    if (cursor.moveToFirst()) {

        do {
            int classID = cursor.getInt( columnIndex: 0);
            String className = cursor.getString( columnIndex: 1);
            String classTeacherName = cursor.getString( columnIndex: 2);
            String classTeacherEmail = cursor.getString( columnIndex: 3);
            int day1 = cursor.getInt( columnIndex: 4);
            int day2 = cursor.getInt( columnIndex: 5);
            int day3 = cursor.getInt( columnIndex: 6);
            int day4 = cursor.getInt( columnIndex: 7);
            int day5 = cursor.getInt( columnIndex: 8);
            int day6 = cursor.getInt( columnIndex: 9);
            int day7 = cursor.getInt( columnIndex: 10);
            String dayTime1 = cursor.getString( columnIndex: 11);
            String dayTime2 = cursor.getString( columnIndex: 12);
            String dayTime3 = cursor.getString( columnIndex: 13);
            String dayTime4 = cursor.getString( columnIndex: 14);
            String dayTime5 = cursor.getString( columnIndex: 15);
            String dayTime6 = cursor.getString( columnIndex: 16);
            String dayTime7 = cursor.getString( columnIndex: 17);
```

```
            UserClass userClass = new UserClass(classID,
                    className,
                    classTeacherName,
                    classTeacherEmail,
                    day1,
                    day2,
                    day3,
                    day4,
                    day5,
                    day6,
                    day7,
                    dayTime1,
                    dayTime2,
                    dayTime3,
                    dayTime4,
                    dayTime5,
                    dayTime6,
                    dayTime7);
            returnList.add(userClass);

        }while (cursor.moveToNext());
    }
    else {
        //No classes are added. This is so application dose not crash if there is no classes.
    }

    //Close cursor and db
    cursor.close();
    db.close();

    return returnList;
}
```

This method iterated through the database getting each row and storing that in a list of UserClass then returning that list.

```
public List<UserClass> getAllClassesOnSetDay(Integer day){
    List<UserClass> returnList = new ArrayList<>();
    String queryString = "SELECT * FROM " + CLASS_TABLE;
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(queryString,  selectionArgs: null);

    if (cursor.moveToFirst()) {

        do {
            if (cursor.getInt( columnIndex: day + 3) == 1) {
```

This class has the same code as the getAllClasses method expent this method has one more line of code that will only select classes that have the column for the day inputed into this table as true.

```
public boolean deleteUserClass(Integer ID){
    SQLiteDatabase db = this.getWritableDatabase();
    String queryString = "DELETE FROM " + CLASS_TABLE + " WHERE " + COLUMN_ID + " = " + ID;
    Cursor cursor = db.rawQuery(queryString,  selectionArgs: null);

    return !cursor.moveToFirst();
}
```

This method deletes the class that ID matches the inputted ID

addAssignment class

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_assignment);

    //Definition
    addAssignment = findViewById(R.id.buttonAddAssignment);
    nameOfAssignment = findViewById(R.id.editTextNameOfAssignment);
    timeDueOfAssignment = findViewById(R.id.editTextTimeDueOfAssignment);
    dateDueOfAssignment = findViewById(R.id.editTextDateDueOfAssignment);
    descriptionOfAssignment = findViewById(R.id.editTextDescription);

    //Back Arrow
    {
        findViewById(R.id.imageBack).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { onBackPressed(); }
        });
    }

    addAssignment.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            UserAssignment userAssignment;
            DBAssignmentTable database = new DBAssignmentTable( context: addAssignment.this);

            try {
                userAssignment = new UserAssignment( id: -1, nameOfAssignment.getText().toString(), descriptio
                database.addUserAssignment(userAssignment);

                Toast.makeText( context: addAssignment.this,  text: userAssignment.getAssignmentName() + " assign
                onBackPressed();
            }
            catch (Exception e){
                Toast.makeText( context: addAssignment.this,  text: "Failed to add assignment!", Toast.LENGTH_SHO
            }
        }
    });
}
```

This method saves all the information inputed into the fields as a UserAssignment class and then inputs that UserAssignment into the User assignment database.

addClass class

```
d1Start.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        TimePickerDialog timePickerDialog = new TimePickerDialog(
                context: addClass.this,
                new TimePickerDialog.OnTimeSetListener() {
                    @Override
                    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                        t1Hour = hourOfDay;
                        t1Minute = minute;

                        Calendar calendar = Calendar.getInstance();

                        calendar.set( year: 0, month: 0, date: 0,t1Hour,t1Minute);

                        d1Start.setText(DateFormat.format( inFormat: "hh:mm aa", calendar));
                    }
                }, hourOfDay: 12, minute: 0, is24HourView: false
        );
        timePickerDialog.updateTime(t1Hour, t1Minute);
        timePickerDialog.show();
    }
});
```

This is the logic that allows for the time to be inputted into the start and end of class time fields.

```
//Try catch to so that app doest crash if unexpected value is entered into UserClass class.
try {
    if (day_1.isChecked()) {day1IsChecked = 1;}
    else {day1IsChecked = 0;}
    if (day_2.isChecked()) {day2IsChecked = 1;}
    else {day2IsChecked = 0;}
    if (day_3.isChecked()) {day3IsChecked = 1;}
    else {day3IsChecked = 0;}
    if (day_4.isChecked()) {day4IsChecked = 1;}
    else {day4IsChecked = 0;}
    if (day_5.isChecked()) {day5IsChecked = 1;}
    else {day5IsChecked = 0;}
    if (day_6.isChecked()) {day6IsChecked = 1;}
    else {day6IsChecked = 0;}
    if (day_7.isChecked()) {day7IsChecked = 1;}
    else {day7IsChecked = 0;}

    day1Time = d1Start.getText().toString() + " to " + d1End.getText().toString();
    day2Time = d2Start.getText().toString() + " to " + d2End.getText().toString();
    day3Time = d3Start.getText().toString() + " to " + d3End.getText().toString();
    day4Time = d4Start.getText().toString() + " to " + d4End.getText().toString();
    day5Time = d5Start.getText().toString() + " to " + d5End.getText().toString();
    day6Time = d6Start.getText().toString() + " to " + d6End.getText().toString();
    day7Time = d7Start.getText().toString() + " to " + d7End.getText().toString();

    userClass = new UserClass( id: -1, editTextNameOfClass.getText().toString(), editTextTeacherName.getText
            editTextTextEmailAddressOfTeacher.getText().toString(), day1IsChecked, day2IsChecked, day3IsCh
            day5IsChecked, day6IsChecked, day7IsChecked, day1Time, day2Time, day3Time, day4Time, day5Time,
    boolean debugSuccess = dataBase.addUserClass(userClass);

    Toast.makeText( context: addClass.this,  text: userClass.getNameOfClass() + " has been added.", Toast.LENGT

    onBackPressed();
}
catch (Exception e){
    Toast.makeText( context: addClass.this,  text: "Unable to add new Class!", Toast.LENGTH_SHORT).show();
}
```

This gets all the data in the field and saves that to UserClass then inputs that into the user class database for it to be saved.