Josh Westenheffer

CS 499

Module 3

Milestone 1

**QUESTION 1:**

The artifact is an Android Studio calendar that initially features a basic login screen, data display screen, and an SMS screen. It was created last August during my Mobile programming class that centered around Android Studio. I wanted to choose this artifact because in my mind it has everything I need to show how I have improved as a coder. It has a login screen that I can improve with a database to show how I can improve security and user experience. It has a data display where I can use a database in tandem with hash maps to show how I can create efficient and unique solutions. Finally, the login and data from the calendar itself is a great way to showcase how I can use a database and create a solid user experience through screens.

**QUESTION 2:**

In this specific milestone I improved on the xml layout file for my data display and the logic in my data display java file. The goal was to add functionality to move through every month in the year while displaying the dates correctly depending on when the month starts. So in a short list here is what I changed:

- Functional Changes
    - Calendar now features all 12 months / Instead of 1
    - Calendar months feature their specific number of days on the correct days
    - Added last and next buttons for cycling through the months
    - Added a back button to the SMS page

- Code changes
    - Added and changed logic in the data_display java file
    - Added to the data_display xml display file
    - Added to the sms_display xml display file

**QUESTION 3:**

The course outcomes I was hoping to achieve were outcome 3 and 4. Outcome 3 is focused on designing and implementing solutions while considering their tradeoffs primarily focusing on data structures and algorithms. For this I went with a hashmap because it is efficient and my data doesn't need to be ordered. Here is a look at the code and I will explain it for this section:

```
private HashMap<String, HashMap<Integer, String>> calendarData;  2 usages
//Create an array named months to be placed in the keys section of the calendData hashmap
private String[] months = {"January", "February", "March", "April", "May", "June", "July",
```

```
calendarData = new HashMap<>();
for (String month : months) {
    calendarData.put(month, new HashMap<>());
}
```

The above code is a look at the hashmap I am using and how it is being utilized to efficiently stores months and in the future data for each month. The first screenshot shows the hashmap being created with the name calendarData, strings are used as the keys and a nested hashmap is used for the values. The nested hashmap takes integers for keys and strings for values. Then I create an array featuring all the months. The second screenshot shows the logic behind initializing the hashmap and placing each value "month" in the array "months" as a key with a nested hashmap in the calendarData hashmap. This has allowed me to efficiently store my months and their days without data. I will include data being stored later which is why the hashmap features the nested hashmap.

The above explanation is an example of how my added code meets the 3rd outcomes expectations by making use of a data structure and designing an effective solution.

Here is a brief informative showcase of the textviews for my xml:

```
<!-- Calendar Grid (Days of the Month) -->
<GridLayout
    android:id="@+id/calendar_grid"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:columnCount="7"
    android:rowCount="5"
    android:padding="8dp">

    <!-- Days 1 to 31 -->

    <TextView
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_row="0"
        android:layout_rowWeight="1"
        android:layout_column="4"
        android:layout_columnWeight="1"
        android:layout_margin="2dp"
        android:background="#ffffff"
        android:gravity="center"
        android:padding="8dp"
        android:text="1"
        android:textSize="18sp"
        android:clickable="true"
        android:focusable="true"/>
```

That is the before picture for my grid text. You can see the comment that there are 31 total, now I wanted to add logic to populate these based on the month so, I added 11 views to the grid so there are 42. This will give me enough space to populate for any given month depending on start date. I don't have an after picture because it is the same code for the textviews and grid, but the weighting for the rows and columns are different. Also, the text in them are blank, but this doesn't matter because this gets populated anyways in the next function I will show.

The second outcome is about using innovative techniques to accomplish goals with focus in software engineering and design. For this I believe the logic I designed for filling in the days of a month when it is switched to should suffice. Here is a look at that code and a short breakdown of it and the outcome:

```
private void updateCalendar() {  2 usages
    // Set the current month name
    String currentMonth = months[currentMonthIndex];
    currentMonthText.setText(currentMonth);

    // The results of these functions will help fill each month properly by i
    int daysInMonth = getDaysInMonth(currentMonth);
    int firstDayOfWeek = getFirstDayOfWeek(currentMonth);

    // I have 42 TextViews written in the xml that this will potentially be p
    for (int i = 0; i < calendarGrid.getChildCount(); i++) {
        View view = calendarGrid.getChildAt(i);

        // Ensures that things like the titles of the weekdays won't be popul
        if (view instanceof TextView){
            TextView dayView = (TextView) view;

            // Since we have the first day and the amount of days in the mont
            // If the grid position is not a valid day of the month leave it
            if (i < firstDayOfWeek || i >= firstDayOfWeek + daysInMonth) {
                dayView.setText("");
            } else {
                // Add 1 because TextView starts with an index of 0
                int day = i - firstDayOfWeek + 1;
                dayView.setText(String.valueOf(day));
            }
```

The code above is the logic for filling in the correct days for a month when the user navigates to that month. The logic takes the current month and uses two other functions that I created, the daysInMonth, and the firstDayOfWeek. In short, the days in month I just used a switch statement and returned the value of that month. The first day of the week I imported Android Studios calendar class which features a Day of month / week variable that takes the years calendar. For this I took the first day of the month and set the first day of week to Sunday and that function returns firstDayOfWeek.

The first for loop is straightforward, it takes the grid of 42 Textviews I wrote in the xml file and will iterate over them. Then to make sure I don't populate textviews like Monday, Tuesday etc. I check if the textview is a part of the grid if not it won't be considered. The final if statement looks to see if the grid textview index (i) is lower than the firstDayOfWeek. The day of week is stored 1-7 and the textviews are stored 0-41. The second part of the if accounts for any days that are outside the bound of total days in a month by adding the firstday value to the value of days in the month. Set the textview label to be blank if either of these are met.

The else covers a valid textview and will fill it with the current i value. Not the +1 which happens because the grid index is 0-41 rather than 1-42.

This block of code demonstrates how I can use innovative techniques to design important software engineering concepts. By doing this I am in turn implementing computer solutions that deliver value and accomplish my goals.

**QUESTION 4:**

I have never interacted with a nested hashmap, so this was a first for me. I only ever heard and saw them in our resources. I am enjoying using it, but it gave me a few challenges like creating the basic array functionality to set it to the keys. It was more that my brain was being overloaded by using the nested hashmap at the same time as the array into the keys which was causing me to become confused. Once I saw it clearly it made so much sense though which was enjoyable.

I also had to interact with functionality for populating the days and that was a beast to deal with. Firstly, I had to sift through the Android Studio import class documentation and find how the calendar import worked. From there I struggled with getting my logic to be correct and get all the days to show on the correct squares. One issue was that I didn't recognize the indexing for the xml was 0-41 instead of 1-42, this would cause days to show up on the wrong numbers. Another issue is that I had an extra day in my xml I had forgotten to delete with a set text of 0. So, when this would get recognized as valid and usable, it would either shift my days or end up in front of a valid day and erase a day like 3 or 2. Working through these makes me appreciate how aware I need to be with what pieces of my code are interacting with what I am working on. I understood what I was trying to populate and what could affect it, but the basic rules and obvious issues still stumped me for a little bit.