Homework 08 – Listview and Internal Storage

Due: Day 32 – April 25th, 2022 (beginning of class)

Objective:

The goal of this assignment is to gain experience with the ListView element and reading/writing from internal storage that we discussed this week.

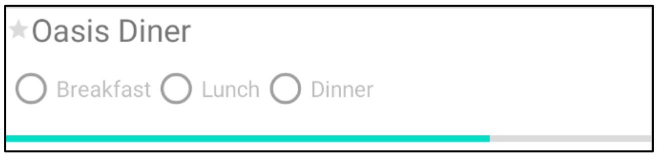
Instructions:

For this assignment, assume that you have been hired by a company to work on its Restaurant Rater app that allows a user to rate restaurants that they visit and store the reviews on their device's internal storage. On Moodle, I have provided a starting Android project that I want you to use for implementing the required features described below. **Important**: The Views in the layout files have already been given an appropriate ID name for this homework. Therefore, do not add/modify/remove any of the Views in the layout files that I provided you. You will need to setup a View Binding in your Java controller files.

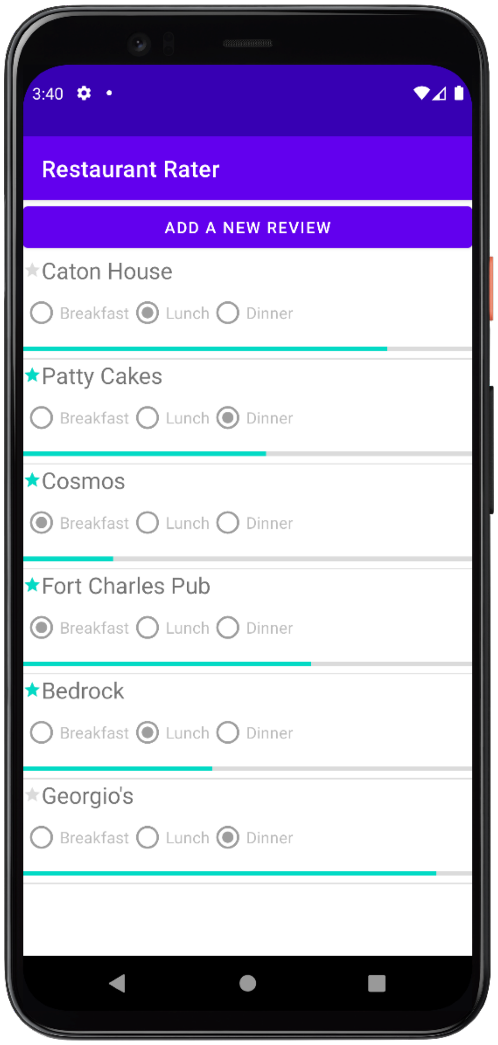
Your app needs to work as follows (i.e., the features/behavior listed below will be explicitly graded for this assignment).

- **Activity # 1 – View Reviews Activity**
    - On Moodle, I have provided a file named **reviews.csv** which is a text file where each column of information is separated by a comma. Download and open up this file with a text editor on your computer to view its contents. This csv file contains an existing list of reviews with the following columns:
        - Column 1 = Restaurant's name
        - Column 2 = Date of restaurant review
        - Column 3 = Time of restaurant review
        - Column 4 = Meal (Breakfast, Lunch, or Dinner)
        - Column 5 = Rating of restaurant
        - Column 6 = Is Favorite (1 = True , 0 = False)
    - Open the starting Android project that I have provided and press the Run button to open the app in the AVD for the first time. Once your app is running in the AVD, open up the **Device File Explorer** tab to navigate the folders/files of the AVD as we did in class. Navigate to your app's internal storage folder on the device (**data >> data >> com.depauw.restaurantrater**). Right-click on the folder named 'com.depauw.restaurantrater' and select **New >> Directory** and enter **files** for the folder name in the window that appears. At this point, you should see three subfolders within your app's internal storage folder: **cache** , **code_cache**, and **files**. Right-click on the **files** folder that you just created and select **Upload**. Then, select the **reviews.csv** file that you downloaded onto your computer in order to upload this file onto the AVD. At this point, you should now see the reviews.csv file inside of the **files** folder that you created
    - Next, you will need to create a **Review.java** class for constructing Review objects to encapsulate each row of the CSV file's data (i.e., each Restaurant Review). Examine the data found on each row of the CSV file to determine what member variables (fields), constructor(s), and getter(s) you will need to add to your Review class/objects
        - Important: Be sure to obey good object-oriented principles as were taught in CSC232 (i.e., member variables should be private, variables and methods should be well-named, etc. – these will be worth points on this assignment's final grade)
    - Afterwards, you should write the necessary code to read in the data in the **reviews.csv** file and parse each line into an array of its individual values. A helpful method to divide a String's data into an "array of Strings" is the split( ) method found in the String class.

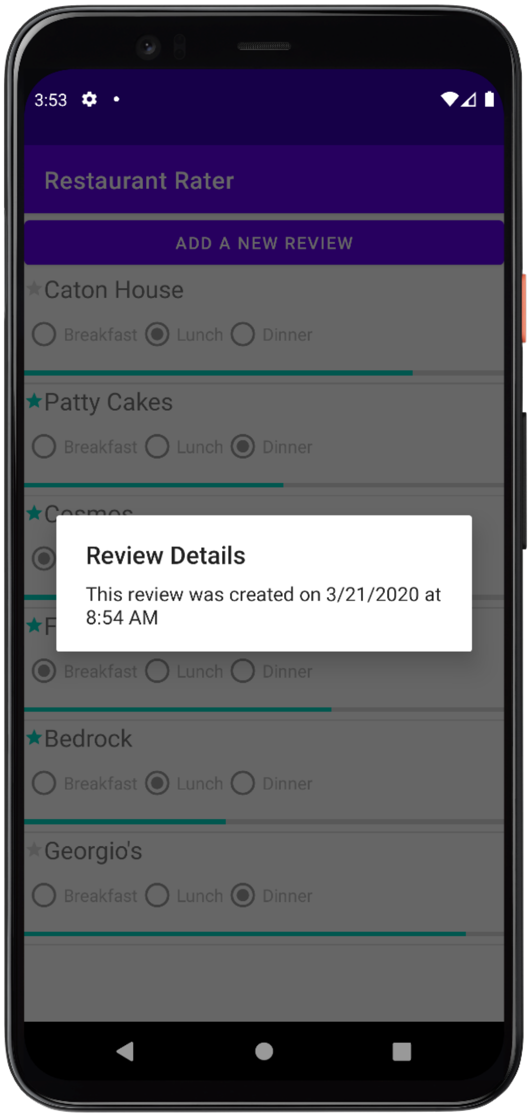
        - Tip: Tackle this problem one step at a time. For example, make sure that your program can read and print each line (String) of the file to Logcat. Then, work on split()-ing the String into an array before trying to extract individual values from the array
- Once you have code to successfully read and parse (i.e., split) each line from the CSV file, you will want to store each row's data into a **Review** object. Then, create a member variable in the activity's class of an appropriate data structure type to store these Review objects.
- Next, you will need to create a layout file for the ListView to display information in the **exact** format shown below. In the top-left corner, the star should indicate whether the review was for a "favorite" or not which is adjacent to the restaurant's name (e.g., Oasis Diner). Beneath this information, a group of Radio Buttons should be displayed for the meal – be sure to set each Radio Button's **enabled** attribute to false so that they cannot be accidentally changed by the user inside the ListView. Finally, beneath these Radio Buttons, a progress bar will visually display the user's rating on a 0 to 100 scale. **Important**: Be sure to give each of these elements a good id name according to the Android Guideline/conventions that we discussed in class (you do not need to give the LinearLayout's an id name). You should use a LinearLayout as the root of your row's layout file and you will need to "nest" LinearLayouts in order to achieve the same format shown below. When you add Views inside of a LinearLayout, be sure to set their layout_weight parameter to **0** before adjusting their attributes.



- Afterwards, add the necessary code for the ListView to display the restaurant reviews from the **reviews.csv** file. When you are finished with this step, your ListView should look as follows:



- Finally, when the user clicks on a specific row/review in the ListView, the company would like a dialog to appear in the **exact** format shown below to display the date and time when the review was created. For example, this was the dialog box that appeared when I clicked on the row/review for **Fort Charles Pub**



Review Details

This review was created on 3/21/2020 at 8:54 AM

- **Activity # 2 – Add Review Activity**
  - When the user clicks the "**Add a New Review**" button on the homepage (i.e., **ViewRatingsActivity.java**), they should be taken to the **AddReviewActivity** page/screen. Add the necessary code to your ViewRatingsActivity.java class to open up this new activity.
  - Once the user is on the Add Review Activity page/screen, they will enter in information for a new review of a restaurant.
  - **Important**: The company wants a DatePicker to appear when the user taps the **edittext_review_date.** The DatePicker should display with the current date initially selected, however, the user should be able to select any date for their review. When the user confirms the review's date in the DatePicker, the date that they selected should be displayed in the **edittext_review_date** in the following format: **month / day / year**
    - Example: If the user selected April 25th, 2022 – it should be displayed as **4/25/2022**
  - **Important**: The company wants a TimePicker to appear when the user taps the **edittext_review_time.** Constructing a TimePicker is very similar to constructing a DatePicker – explore the TimePickerDialog class for a list of its constructor and methods. The TimePicker should display with the current time

initially selected, however, the user should be able to select any time of day for their review. When the user confirms the review's time in the TimePicker, the time that they selected should be displayed in the **edittext_review_time** in the following format:
hour : minute AM or **PM**

    - Example: If the user selected two o'clock in the afternoon – it should be displayed as **2:00 PM**

- Finally, when the user presses the **Add Review** button, the information that the user entered on this activity's form should be added to the end of existing restaurant reviews in the **reviews.csv** file found in internal storage (note: do not re-write all of the data to the same/new file as this will be worth 0% credit). Afterwards, you should call the **finish**( ) method to end (destroy) the Add Review Activity.
- **Very Important**: After the user is returned to the previous activity object (i.e., View Reviews Activity), you will likely notice that the ListView does not display the newly added review – if your code is in the onCreate( ) method, think about why this is happening given our understanding of when lifecycle methods are called by Android. Add the necessary code to "reload" the data from the CSV file into the ListView – note: if you use this approach, you should not copy-and-paste the code that you have so far but rather put this code into a **method** and call the method where you need to execute it.
    - **+5% Extra Credit Opportunity**: As you might imagine, "reloading" the ListView by reading in the CSV file again is rather costly. Therefore, think about how you might instead transmit the new review's data back to the View Reviews Activity after it has been added to the CSV file – note: in order to receive the extra credit, you are not allowed to open up a new instance of the View Reviews Activity.

Submission:

When you are finished, you must **zip** your Android Studio project. On Moodle, you should upload your zip file to the Homework 08 assignment box