Assignment #4

Mobile Development
Due: December 6, 2017

Assignment #4 is a continuation of assignment #3. Below you will find the assignment #3 description in regular text. The assignment #4 description has been added in **bold script**. It has been placed in bold script so that it can be kept in context with assignment 3 and it make it easy for you to see the new assignment 4 additions to the application. Enjoy.

Assignment #3 and #4 will form a single large application. **Please view this assignment as Part 2**. The application we will be building is called a Personal Data Assistant. In this assignment, we will build the general structure of the application and some of the easier features. **We will continue developing this application in assignment #4 adding advanced capabilities**. If, for some reason, you cannot complete assignment #3 a solution will be provided before we start assignment #4. Anyone can use the provided solution as their base for assignment #4, or they can use their own assignment #3 as the base for assignment #4. Assignments #3 and #4 will have an optional Glory section for those of you who are bored and want something a little more challenging.

Our PDA (Personal Data Assistant) will have the following features: (1) A to-do list by project, (2) work hours by project, (3) an expense logger by project, and (4) an invoice creator by project and date range.

In assignment #3, we are only creating the user interface. This includes the Views and Controls, possibly some Models. In assignment #4 we will create the database and reports. **In assignment #4 we will be saving the data we enter into the SQL Lite database.** Assignment #3, however, must use Activities, Fragments, the Action Bar, Menus, Graphics, and Audio. **Assignment #4 must use application storage, file storage and SQL Light, and it must make all the features in the PDA functional. Assignment 3 created the views, while assignment 4 will make the views function.**

Use the JukeBox and ToDo apps we saw in class as example programs for this assignment. Take care to make your app look professional. I give you freedom to decide the look and feel of many parts of your app, however I will insist on certain things.

Your app will begin by presenting a Welcome screen that displays the title of your application, a picture, and a button that says Enter. Pressing the Enter button displays the main screen.

The main screen will use graphic buttons and the Action Bar menu. The buttons will say: TO-DO, WORK HOURS, EXPENSES, INVOICE. The Action Bar menu will have: SETTINGS, HELP, and EXIT. Pressing any of the graphic buttons or menu items will switch to another Activity (except for EXIT). Each button activates its own Activity. In the case of the EXIT menu option, a popup prompts the user by asking “Are you sure?”. The user must answer YES to exit or NO to stay in the program. The Action Bar menu is persistent throughout the entire program. All Activities will have the same Action Bar menu items. You must define this menu only once and share it with all the Activities, however, the HELP option will be sensitive to the currently active Activity, displaying help for that View.

Described here is every button and menu option:

TO-DO

This activity uses three fragments. The activity is divided into two areas: the top half of the screen and the bottom half of the screen. In the top half of the screen we always see the same fragment, the "top fragment". Depending on what happens in the top fragment the bottom part of the screen displays one of two additional fragments. The top fragment displays a dropdown list and a text box, plus three buttons: New Project, New Task, and Main Menu.

Using the Application Storage, the dropdown control remembers what was previously selected. This means, when the user goes away from the Activity and later returns, or closes the application and then comes back, the dropdown displays the previous selected item.

The dropdown control contains the names of all the projects. **The projects are retrieved from the Projects table (see below).** If the user selects a project from the list the bottom fragment displays the tasks belonging to that project (for ass#3 it only switches to the correct fragment). The bottom fragment is the Task List fragment. **In assignment 4 the bottom fragment is populated with all the to-do items from the SQL Light table named TODO. The fields of this table are the following: Int projectID, Boolean completed, String task, and DateTime due. The date time can be left unselected. The completed field is by default false. On the View the completed field is displayed as a checkbox. The task field is displayed as a text box. The DateTime field is implemented using a date picker. The projectID field is not displayed but is automatically populated by the program.**

Pressing the New Project button, changes the lower fragment to the Create Project fragment. A textbox and a single button labeled Created Project is present. The user will enter the project name and then press the button to create that project, if it does not already exist. Then the fragment switches to the Task List fragment showing the tasks for that newly created project (which should be empty since it was newly created). **The Create Project fragment uses SQL Lite. A table named Project exists with the following fields: int projectID and String projectName. The field project ID is a unique number, it cannot be left empty. The program automatically populates this value using the SQL auto increment feature.**

The Task List fragment displays a list of checkboxes that are either checked or unchecked by the user. Beside each checkbox is the task and a Delete button that the user can use to remove that task from the list. For ass#3 make this functional. **In ass#4 you will save and restore this information from the database**. **See the above table description for TODO. Notice that TODO has a projectID field that must match with the Project table's projectID field for those tasks that belong to the associated project.**

The New Task button, from the top fragment, when pressed, will add the text from the textbox into the Task List of the project currently selected in the dropdown. **It will also create a new record and populate its fields for the TODO table.**

Pressing the Main Menu button, from the top fragment, switches the Activity returning the user to the main menu Activity. **In ass#4 you will need to make sure all the data is saved and database is closed.**

WORK HOURS

Think of this Activity as being similar to an employment punch-in punch-out program, but instead of punching in you are simply **recording the time you used up to finish** some task.

This Activity is a single screen populated by a dropdown, a textbox, a start time control, an end time control, a list box, a button labeled Record Time, and a button labeled Main Menu. The dropdown is populated by the projects created from the **Projects table**. The list box is populated by the hours recorded for the selected project. **An SQL Light table called HOURS exists with the following field: int projectID, DataTime start, DateTime end, and String description.** Selecting a project from the dropdown automatically displays the hours associated with that project in the list box. The user adds new records to the list by selecting a start time / date and an end time / date, **and writes a description of the work performed** and then pressing Record Time. The user can record hours multiple times (multiple records). The textbox is the description of the work performed during that time period. **The HOURS table is sorted by start time, lowest to highest, and is grouped by projectID.**

Pressing the button Main Menu returns the user to the main menu activity.

We won't implement a delete button.

In ass#3 you are only making the View functional. **In ass#4 you will be building the database and populating the controls from the database.**

EXPENSES

The EXPENSES feature is like WORK HOURS. I will not repeat the description, see above.

Instead of start and end times, the activity displays a textbox for the description of the expense and a textbox for the expense amount. The expense amount must be a positive real number.

An SQL Light table is used to record the information, called EXPENSES. It has the following fields: int projectID, DateTime date, String description, float amount. The program automatically populates the date field with the current date and time when the record was recorded. The table is sorted by date (lowest to highest(and grouped by projectID.

INVOICE

The Invoice feature is made from two Activities. The first activity is a form asking the user to select a project from a dropdown **(loaded by values from the Project table)**, a date range (start date to end date, no hours), and a button labeled Create Invoice. The dropdown displays the project names created from the To Do List activity. When the Create Invoice button is pressed, the application switches to the second Activity.

The second Activity is the invoice report. Design a professional looking invoice. For assignment #3 the application simply switches to the second activity. **In assignment #4 the code for generating the report will be developed. The information for this invoice will be takes from the**

tables Project, Hours, Profile (see below) and Expenses. The invoice will have a header portion at the top left, details in the form of a table in the middle, and the taxes and total at the bottom right. The table could be a grid view or in rows, however the columns must be aligned.

The top left will display: the user's full name (see Profile below), project name and today's date. On the right side is displayed the invoice number, taken from the Profile table.

The middle will display row-by-row, first, all the hours worked and then all the expenses that are associated with the selected project and within the selected date range. Each row will display the start date/time, and end date/time (if available), description, hourly rate (see Profile table from Settings), number of hours (if available) and then the extension amount. The extension amount = (hours * rate) + expense. In the case of the Expenses table there is no hours or rate, so the extension column will simply display the total expense.

At the end of the invoice screen is a button that says SAVE. Pressing it causes the program to create a text file in your internal storage named: invoice_NO_DATE.txt, where DATE is the current date and time when the user pressed the save button, and NO is the invoice number. If the user presses the save button then the invoice number, from the Profile table, is incremented, so that the next invoice will have a different number.

SETTINGS

It contains a Profile option. This option displays an additional view connected to an SQL Light table called Settings. This table's fields are: String firstName, String lastName, float payRate, and int invoiceNumber. The invoice number is initialized to 1 when the table is first created. The view has a title that says Settings. The View only supports a single user. Your table will contain only a single record.

HELP

Is a single page that displays help to the user for the current activity. The information displayed in the help screen is sensitive to the feature the user is in.

GLORY SECTION

This section is optional. **In assignment 4 you can make these glory questions functional.** You do not need to do it. It is provided for those students who want to do more. These glory options are not for points. They are for bragging rights only. You do not need to do all the glory options, just the ones you like. Here are the options you can add to this assignment:

- For the To Do List, ordering your to-do items is useful, so provide a way for the user to be able to move items up & down.
- For Work Hours, provide a way the user can generate a text file reporting on the hours they worked for a particular project.
- For Work House, create an additional report that when the user does not provide a project name it generates a report sorting all the project names and displaying their total hours per project.
- For Invoice, provide a way to display a bar graph reflecting the monthly sales total for a period of time.

GRADING

This assignment is out of 20 points

To Do	5 points
Work Hours	3 points
Expenses	3 points
Invoice	5 points
Settings	3 point
Help	1 points (for Profile)