**James Cook University Singapore**

**CP3406 Mobile Computing – Practical and Assignment Guidelines**

**Week1 - MyDiet**

Based on the Welcome IT app, its walkthrough and discussions, make an app called “MyDiet” that displays a healthy food and an inspirational message.

**Extra:**

* Do the [build your first app](https://developer.android.com/training/basics/firstapp/index.html) google training exercise using Android Studio.
* Each time the “MyDiet” app runs:
* randomly displays an image from a collection of food related images (at least 3)
* inspires the user with randomly selected motivational words of wisdom (at least 3)

**Week2 - QuickSum**

The purpose of the “Quicksum” app is to provide fast and accurate way of adding values. It’s not a general-purpose calculator. It’s meant as a marking tool that educators and teachers can use when they are marking quizzes or exams.

**Requirements and Behaviors:**

* The app has a main activity only
* The main activity contains a label for displaying the calculated sum and buttons for values 1,2,3,4,5,6,7,8,9 – pressing these buttons adds their value to the sum.
* The main activity has another button called "other" that swaps the labels of buttons 1,2,3 to be 1/2, 1/3, ¼
* As soon as one of these buttons is pressed, its value is added to the sum, and all of these buttons return to 1,2,3 immediately

**Extra:**

* Create a second activity
* Use this as a look-and-feel configuration for the app
* Provide options for changing the background color for the main activity
* E.g. provide a set of color choices - selecting one sets the background color

**Week3 – GuessingGameApp**

Make a guessing game app. This app has a main activity for the game, and (extra) a secondary activity for configuring the game.

**Requirements and Behaviors:**

Both activities must:

* Support portrait orientation only
* Use the same custom material theme (choose your own colors, etc.)

The main activity has:

* A vertically-oriented linear layout that contains:
  + A horizontally-oriented linear layout
  + A button for navigating to the secondary activity
  + The inner linear layout contains a TextView for displaying the status of the game and an EditView that only accepts digits
  + Use TextWatcher to handle changes in the EditText

The (extra) secondary activity consists of:

* A pair of nested linear layouts, or a 2x2 grid layout containing a two SeekBars and two TextViews - these are used to set the range of numbers for the game (i.e. the minimum and maximum values in the range).
* A button for closing the secondary activity and resuming the main activity.

**Behaviors:**

The main activity behaves as follows:

* When it enters the foreground, a random number is chosen between the minimum (default 1) and maximum numbers (default 10).
* The user is able to enter a guess into the text field.
* If the guess doesn’t match the randomly chosen number, then the status message is used to tell the user to “try smaller” or “try bigger”.
* If the guess matches the randomly selected number, then the status message is used to tell the user they have won.
* When the button is clicked the app navigates to the secondary activity.

The secondary activity behaves as follows:

* The user is able to set the minimum and maximum numbers using the seek bar
* When the button is clicked the app returns to the main activity.

**Extra:**

* find a way to maintain the current minimum and maximum numbers. So if the app is stopped and restarted, then the most recently selected minimum and maximum numbers are used by the game.

**Week4 - GuessTheCeleb**

**Technical purpose:**

* Dynamically apply fragments to an activity depending on device orientation.

**Requirements and Behaviors:**

* The app must include two fragments (check out the support screencast with a simple demo)
* A MainFragment for the game play.
* A SecondaryFragment - for game configuration.
* When the app is in portrait mode, the MainActivity shows the app toolbar and the MainFragment - the app toolbar contains a settings button that opens up a SecondaryActivity that contains the SecondaryFragment.
* When the app is displayed in landscape mode, both fragments are visible side-by-side - and the app toolbar must not be visible.
* The MainFragment needs an ImageView and a GridView of items (2-6).
* Use the ImageView to display the face of a famous person.
* The items should the names of different famous people.
* The player wins by selecting the correct name for the currently displayed famous person from the list of names - use a Toast to inform the user when they are correct or incorrect.
* Use a set of at least 5 famous people in your game.
* The SecondaryFragment is used to select the number of names available in the GridView during gameplay.

**Week5 – PhotoEditor**

**Create an** **app** **that allows the user to select an image from the device photo gallery, and then the user is able to draw over the image using simple 2D drawing (at least support black** **colored, 1pixel wide line drawing).**

**Extra:**

* Support different colors and drawing widths. Use fragments - support small width and large width devices - on a small width device drawing settings are accessed via the action bar which leads to a settings activity, on large width devices the drawing settings and the drawing itself are visible side-by-side.

**Week6 – Assignment One**

Develop an Android utility app for an activity useful in daily life.

**Requirements and Behaviors:**

* You are free to choose which activity and what aspect of daily life (e.g. utility app for the current weather right now where you are, or how about a unit conversion app for a commonly used measure such as $USD to $AUD).
* To be classified as a “utility app”, your app must clearly display the most important elements of your chosen daily life activity in a way that is easy to understand and easy to use.
* Your utility app must consist of:
  + A main screen for displaying the content or functionality.
  + A setting screen that controls some aspects of the content or functionality.

**Other Requirements:**

* The coding style you use must be clear and easy to follow - naming conventions should be consistent and support the readability of your code, comments are useful for complex parts of your codebase.
* The code should be well formatted - you can use Android studio to help you with this!
* You are expected to make use of the unit testing features in Android studio - e.g. for a unit conversion app, you might write some unit tests that verify your conversion calculation method works for different input values.
* You use of XML in your Android studio project work should be in-line with the standard way of applying it as described in the subject materials, in terms of the project basic manifest file, basic app permissions, basic resource files, and basic activity layouts.
* Standard use of touch events as described in the subject materials is expected - how intuitive the gestures are for the user is an important consideration.
* Basic use of Activity class lifecycle methods is expected for the main screen and the settings screen.

**Week7 - DrumKitApp**

**Using SoundPool create a full screen app that present the user with a picture of a drum kit. When the user touches a drum/cymbal in the drum kit the corresponding sound for that drum/cymbal is played.**

**Requirements:**

* Create this app using the MVC design pattern. So, that means creating a utility class for managing the sound pool, loading its audio files, when all audio files are loaded then sounds can be played.

**Extra**:

* Add a shared preferences activity using the PreferenceFrament coding style (See part 1 of Flag Quiz Demo). Allow the app to support small screen size portrait mode, and large screen size landscape mode.

**Week8 - SpiritLevelApp**

Create an app that acts like a [spirit-level](https://en.wikipedia.org/wiki/Spirit_level). The app must use the **metaphor** UI design pattern.

**Method:**

* Use the x-axis, y-axis raw data from the accelerometer sensor. There is no need to use the z-axis data - it points through the screen of the device. Moreover, there is no need to convert the raw data into angular data. Simply use the fact that the "max" value for gravity is 9.8. It is advisable to limit the floating-point accuracy of the raw data - since it tends to move randomly at small values. Use (perhaps) one or two decimal points only

**Extra:**

* Use the gravity sensor instead of the accelerometer sensor on devices that support a gravity sensor. Ensure that the sensor data is enabled and disabled only while the app is in the foreground. If the app were to be placed on Google Play Store, then the only devices able to install it are those that support accelerometer data.

**Week9 – Assignment Two**

Create an educational game targeting high-school students (years 7-9) that supports STEM learning.

**Instructions:**

The app consists of **four** screens:

* A main menu or landing page
* A game screen
* A setting screen
* A leaderboard screen

The game is expected to support **one** aspect of STEM (Science, Technology, Engineering, Mathematics). Some examples are:

* The solar system
* The natural environment
* Environmental issues (e.g. how about a game that promotes awareness of nature disasters?)
* Essential knowledge about computers/mobiles/web
* Basic puzzle games (calculate the answer to a series of math questions)

**Requirements:**

* Coding style should be appropriate (use of MVC, utility, DAO, etc.).
* Apply appropriate GUI design patterns through the app screens.
* Make it a "full-screen app" with a GUI design that's intuitive, clear, responsive, and visually interesting.
* Use an "action bar" to provide convenient access to the settings and high scores
* Use age-appropriate graphics, background music, and sound effects to help make the learning experience fun.
* Allow the app to post updates to a Twitter feed, or other similar use of social media (based on settings preferences).
* Also include the use of sensor data (e.g. accelerometer data) as a mechanism to control game-play.
* The app should include puzzles that rely on some combination of task memory, pattern recognition, and the promotion of patience while learning (pick at least **one** of these things).
* Deploy and promote your app effectively on the Google Play store.
* Use a well-designed manifest file, create the minimum required promotions graphics.
* To support the high scores screen, your app needs to record some measurable quantity (e.g. time taken to solve the puzzle).