CST2335 Graphical Interface Programming

# Final Project Assignment

Although this is a group project, you must only work on your part of the project that you have selected. You must write all of your own code that makes up your part of the project. If you use code that someone else, including your teammates, then that is considered plagiarism, and you will get a mark of 0 for the final project.

## Purpose

The Project will give you experience in:

* Developing software in a group environment, including using GitHub to merge code into 1 project.
* Dividing a workload to meet deadlines.
* Designing modular software that allows for that division of work.

## Part 1 – Choosing your team

* + Task Done
  + This is worth 5% of your final project mark

## Part 2 – Programming your application.

## 2.1 Requirements

Once you have chosen your topic, you must implement this list of the requirements for the final project:

1. All activities must be integrated into a single working application, on a single device or emulator. You should use GitHub for merging your code by creating pull requests. The demo must occur from the **main** branch on your project’s repository.
2. The entire project must have at least 1 activity written by each person in your group (so at least 1 activity per person). Your activity must be accessible through a graphical icon from a Toolbar.
3. Each person’s project must have a RecyclerView somewhere to present items in a list.
4. Each person’s project must use a database to store items. The user must be able to add and delete items, which should be displayed somehow in a RecyclerView. It should work similarly to the chat room labs you did this semester.
5. Each activity must use Volley to retrieve data from a server. You cannot use Executor or AsyncTask
6. Each activity must have at least 1 Toast, 1 Snackbar, and 1 AlertDialog notification.
7. Each activity must have at least 1 edit text with appropriate text input method and at least one button.
8. Each activity must use SharedPreferences to save something about what was typed in the EditText for use the next time the application is launched.
9. Each person’s project must have a help menu item that displays an AlertDialog with instructions for how to use the interface.
10. There must be at least 1 other language supported by your part of the project. Google Translate can help you perform the translations.
11. The interfaces must look professional, with GUI elements properly laid out and aligned.
12. The functions and variables you write must be properly documented using JavaDoc comments. Each file must include a header stating the purpose of the file, the author, the lab section, and the creation date.
13. Each team member must add at least 5 test cases that test the interface on their part of the application.

## 2.2 Beginning Steps

* One person should create a new project for the team and then upload it to github using the menu option “VCS” -> “Import Into Version Control” -> “Share project on GitHub”.
* That group member must then invite the other group members to contribute. This is done by clicking on the “Settings” tab in Github, then click “Collaborators” on the left side menu and search the group member names to add them to the project. Other team members should then clone that project to their computer and start making branches for their work.
* From AndroidStudio, select “File” -> “New” -> “Project from version control” -> “Git” and then paste the git URL from the main github repository from the previous step. ***You will not be able to integrate your work if you do not start by first cloning the project!***
* Then write your own code on your own branch and then merge that branch on Github
* You should merge your code to the **main** branch on a regular basis during the development of your application. Don’t try to merge the code on the last week.
* The demo must happen from the **main** branch.

## 2.3 Grading Guide

1. Each student is graded on his or her application separately. *This counts for 85% of your project mark.*
2. Week of April 1 – April 5- Project demonstration during your scheduled lab demonstration found on the demonstration schedule.
3. You will show each of the 13 requirements from the list on the main branch of your repo.
4. Arrange a single submission of the group deliverable by one of the group member’s computer on behalf of the entire group.
5. You must be in the lab in person to answer questions about your work.
6. Code submitted on Brightspace will not be marked.

**2.4 Warnings**

1. Your project will be marked zero if:
   1. The code does not compile
   2. The application crashes on startup or randomly during execution
2. Penalties will apply if:
   1. There are hard coded string literals in the layout/menu xml files
   2. Java code contains string literals except for Log statements

## **2.5 Submit your source code on Brightspace**

* Before your group demonstration, each member of the group must submit their final code as a record of what was finished at the end of the project. From your github repository, there is a link for “Clone or Download”. Select the Download option and save your code as a zip file on your computer. Then upload that zip file to Brightspace using the FinalProject link. ***This is worth 10% of your final project mark.***

**Part 3**

## 3.1 The Application Topics

Each of the applications (as they are intended) requires similar programming techniques. Each application takes information from the user and stores it in a database. They can then view the data saved to a list of favourites and delete items from that list. Beyond that you are free to get creative.

**Sunrise & Sunset lookup**

* Your application should have an EditText for entering the latitude and longitude of a location where you want to look up when the sun will rise and when it will set. The user can then click on a “Lookup” button and the app will send a query to the server.
* The URL for searching is “[api.sunrisesunset.io/json?lat=XXXXX&lng=YYYYYY&timezone=UTC&date=today](https://api.sunrisesunset.io/json?lat=38.907192&lng=-77.036873&timezone=UTC&date=today) where XXX is the latitude of the location and YYY is the longitude of the location.
* Your application should show the results from the server, and also have a button to save that location to the database.
* There should be a button to view “favourite” locations from the database in a recycle view. If the user clicks on one of those locations, then the application should then do a new query for that location using today’s date to show current times for sunrise and sunset.
* There should also be a button to delete that favourite location from the database.
* The SharedPreferences should save the user’s search term so that the next time you start the application, the previous search term is shown in the search field.

**Recipe Search**

* Your application should have an EditText for entering the name of a recipe to search for. There should be a button for the user to send the query to the server and get a list of results back.
* The URL for searching is “https://api.spoonacular.com/recipes/complexSearch?query=XXXXX&apiKey=YYYYY” where XXXX is the recipe to look up, and YYYY is your api key.
* You can get your own api key from here: <https://spoonacular.com/food-api/console#Dashboard>
* The server will return an array of recipes that match the search. Those should be displayed in a recycle view and if the user selects one from the list, you should show the details of the recipe including: Image, Summary, and the “Spoonacular Source Url”
* The original query will return a Title, Image, and ID. Use this second query to get the details for that recipe by looking up the ID: <https://api.spoonacular.com/recipes/> ID /information?apiKey=YYYYY. This second query will give you the Imge, Summary and the source URL.
* There should be a button in the details to save this information to the database. The user should be able to view a list of recipes saved in the database for viewing in the list of recipes. When the user selects from the list of saved recipes, then again show the details but this time there should be a button to remove this recipe from the database.
* The SharedPreferences should save the user’s search term so that the next time you start the application, the previous search term is shown in the search field.

**Dictionary API**

* Your application should have an EditText for entering a word to look up the definition.
* The URL for searching definitions is <https://api.dictionaryapi.dev/api/v2/entries/en/XXXX>. but replace XXXX with the word to search. The server should return an array of possible meanings. You should display these definitions in a recycle view.
* The user should be able to save the search term and the definitions in the local database for viewing later.
* There should be a button for the user to view saved search terms, and selecting one of them should show the definitions for that word. However when viewing a saved definition, there should be a way of deleting definitions from the database.
* The SharedPreferences should save the user’s search term so that the next time you start the application, the previous search term is shown in the search field.

**Deezer song search api**

* The user can search for artists using the api: <https://api.deezer.com/search/artist/?q=XXX>, but replace XXX with the artist that the user enters.
* The results from the server have a field called Tracklist=, with a second URL of all their songs on that album. You should then call that URL to get a list of all the songs. This list of songs should then be shown in a list to the user.
* Clicking on a song should show the title, duration, album name and album cover in a details page. This details page should have a button that saves that song data in a database for later viewing.
* There should be a button that lets the user see the list of favourites saved in the database on their device. Selecting an item from the list of favourites should show the details of the song as mentioned above, and there should be a delete button to remove that song from the database.
* The SharedPreferences should save the user’s search term so that the next time you start the application, the previous search term is shown in the search field.