CST2335 Graphical Interface Programming

# Final Project Assignment

This final project is meant to be an assessment of your work in the lab assignments. You should be able to complete this project without asking for help from the instructor since this is just repeating the knowledge you have learned in the labs. Although this is a group project, you must only work on your part of the project that you have selected. You must write all your own code that makes up your part of the project. If you use code that someone else, including your teammates wrote, then that is considered plagiarism, and you will get a mark of 0 for the final project. Please note that there are four individual projects to be done by each of the four students in the group (one topic per student).

## Purpose:

The Project is assigned to give you experience in:

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

## Part 1 – Choosing your team

This is a group project for groups of 4 members and can be from any lab section within the same theory group in this course. If you would like to be assigned to a group, then send an email to your lab instructor. If you have chosen partners yourself then please email the names of the group to your lab instructor. Once you have a team for the final project, each member of the group should select a different one of the given topics. You should then fill out the Group Activity Worksheet together to exchange contact information amongst your group (Algonquin student email addresses at a minimum). EACH MEMBER of the group must then upload the Excel worksheet on Brightspace, under the FinalProject assignment link on or before ***March 7th, 2025. This is worth 5% of your final project mark***

## 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 (after each requirement is finished). Don’t try to merge the code only on the last week.

## Part 2 – Programming your application. Requirements:

In your group project, you will have a main page that shows once your application is launched. From this page, there should be 4 buttons that launches a page that is the landing page for your part of the project. In your part of the project, you must implement the following requirements:

1. You must have a ListView that lists items that were inserted by the user.
2. There must be a TextField along with a button that lets the user insert items into the ListView.
3. You must use a database to store items that were inserted into the ListView to repopulate the list when the application is restarted.
4. Selecting items from the ListView should show details about the item that was selected. On a phone would use the whole screen to show the details but, on a Tablet, or Desktop screen, it would show the details beside the ListView.
5. Each page must have at least 1 Snackbar, and 1 AlertDialog to show some kind of notification.
6. Each page must use EncryptedSharedPreferences to save something about what was typed in the TextField for use the next time the application is launched.
7. Each person’s project must have an ActionBar with ActionItems that displays an AlertDialog with instructions for how to use the interface.
8. There must be at least 1 other language supported by your part of the project. If you are not bilingual, then you must support both British and American English (words like colour, color, neighbour, neighbor, etc.). If you know a language other than English, then you can support that language in your application and don’t need to support American English.
9. All the individual applications 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.
10. The interfaces must look professional, with GUI elements properly laid out and aligned.
11. The functions and variables you write must be properly documented using Dartdoc comments. You must create the Dartdoc in a Dartdoc folder like you did in the labs.

## Grading Guide

* Each student is graded on his or her application separately. ***This counts for 85% of your project mark.***
* Week 13 All code is merged on your group’s GitHub repository You should only be doing merge fixes and fixing crashes the final week.
* Week 14 of the semester - Project demonstration during your scheduled lab demonstration found on the demonstration schedule. You will show each of the 11 requirements from the list. Arrange a single submission of the group deliverable by one of the group member’s computers on behalf of the entire group. You must be in the lab in person to answer questions about your work. Code submitted on Brightspace will not be marked.
* During the demo, each team member will need to explain their portion of the work
* The demo must occur with the code from the main branch. Marks will be lost for failing to do this.
* Each group will need to show evidence of Git commits, pushes and pull requests in GitHub that shows the contribution of each team member

## Part 3 – 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.***

## The Application Topics

Each of the applications 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. There are 4 topics listed below so each member of your group should choose a different topic to implement the requirements listed in Part 2 above.

**Event Planner page**

This page will be for adding new event to a database of events. You will be able to add, view, update, and delete events.

* Your application should have a button for adding a new event. When the user presses this button, there is a page that lets the user enter the event name, date, time, location, and description. You should check that all fields have a value before letting the user submit the new event.
* Once an event is added, it should appear in a list of events, as well as be inserted in a database. Selecting an event from the list should show the event’s details in the same page as when creating the event. However, instead of a submit button, there should now be an Update and Delete button. The update button just saves the event’s updated information and delete removes the event from the list and database.

When a user adds a new event, the user should have a choice to copy the fields from the previous event or start with a blank page. You should use EncryptedSharedPreferences to save the data from the previously created event in case they want to copy the information.

**Customer list page**

This page will be for adding new customers to a database of customers. You should be able to add, view, update or delete customers.

* Your application should have a button for adding a new customer. When the user presses this button, there is a page that lets the user enter the customer’s first and last name, address, and birthday. You should check that all fields have a value before letting the user submit the new customer.
* Once a customer is added, it should appear in a list of all customers, as well as be inserted in a database. Selecting a user from the list should show that customer’s details in the same page as when creating the customer. However, instead of a submit button, there should now be an Update and Delete button. The update button just saves the customer’s updated information and delete removes the customer from the list and database.

When a user adds a new customer, the user should have a choice to copy the fields from the previous customer or start with a blank page. You should use EncryptedSharedPreferences to save the data from the previously created customer in case they want to copy the information.

**Expense Tracker page**

This page will simulate a user who wants to track their personal expenses. You should be able to add new expense record to list of expenses, view current expenses, update or delete expenses.

* Your application should have a button for adding a new expense record. When the user presses this button, there is a page that lets the user enter the expense name, category, amount, date, and payment method. You should check that all fields have a value before letting the user submit the new expense.
* Once an expense is added, it should appear in a list of all expenses, as well as be inserted in a database. Selecting an expense record from the list should show that expenses’ details in the same page as when creating the expense. However, instead of a submit button, there should now be an Update and Delete button. The update button just saves the expense’s updated information and delete removes the expense from the list and database.

When a user adds a new expense, the user should have a choice to copy the fields from the previous one or start with a blank page. You should use EncryptedSharedPreferences to save the data from the previously created expense record in case they want to copy the information.

**Vehicle Maintenance page**

This page will simulate a car owner or fleet manager tracking the maintenance records of their vehicle(s). You should be able to add a new maintenance record every time a vehicle is taken for service. It should also be possible to view, update and delete records.

* Your application should have a button for adding a maintenance record. When the user presses this button, there is a page that lets the user enter vehicle name, vehicle type, service type, service date, mileage, and cost. You should check that all fields have a value before letting the user submit the maintenance record.
* Once a maintenance record is added, it should appear in a list of all maintenance records kept by the car owner or fleet manager, as well as be inserted in a database. Selecting a maintenance record from the list should show the maintenance details in the same page as when creating the record. However, instead of a submit button, there should now be an Update and Delete button. The update button just saves the record’s updated information and delete removes the record from the list and database.

When a user adds a new maintenance record, the user should have a choice to copy the fields from the previously created record or start with a blank page. You should use EncryptedSharedPreferences to save the data from the previously created record in case they want to copy the information.