# Mobile Computing 2024

Project

## Overview

In this project, you will be tasked with developing a Budget Tracker Android application. The Budget Tracker app aims to empower users to take control of their finances by providing them with tools to monitor their expenses, set budgets, and gain insights into their spending habits over time. As a user you should be able to manually add all your expenses and view them in a list. Each expense should contain a name and an amount. Expenses should be modifiable (edit any part of the expense and delete it) and persistent across app restarts.

## Core Capabilities

Your budget tracker app should include the following capabilities

1. Create expense entries
2. View, edit and manage expense entries within the app
3. Store and retrieve expense entries locally

## Coursework 1

The first coursework involves the design of the app

1. Visual design using wireframes
2. Use case design using UML

- **Wireframes**: Create approximately a dozen wireframes for your app including both core and extended functionality using one of several free online wireframing tools
- You are free to choose the tool of your preference
- We have often used Baslamiq with great results (30-day free license)
- **UML**: Create three UML Use Case diagrams modelling the interactions between your app and its actors
- Submit to the Moodle Dropbox created for Coursework 1 a single PDF document which includes Wireframes and UML

# Coursework 2

The second coursework requires the implementation of the app

## Implement

1) **Core functionality**:
    a) Create expense entries;
        i) Each expense entry must have
            (1) Title
            (2) Amount
            (3) Date of expense
    b) View, edit, and manage expense entries within the app;
        i) All expenses must be visible in a list within the app
        ii) On expense click, user must be able to edit and delete the expense
    c) Store and retrieve expense entries locally
        i) Expenses must be saved locally in a Room database
2) **Extended functionality**: Implement one additional feature of your choice

## Indicative options for extended functionality

- Add a set of categories on the Expenses (e.g. food, shopping, house, transportation etc) and provide messages to the user on the total they have spent on each category.
- Organise your expenses by date and by amount (user can toggle between the two options)
- Setup monthly budget and when the budget is reached notify the user

## Submission

1. All code and supporting material via GitHub Classroom, and
2. zip file export of your GitHub Coursework 2 repo via Moodle

## Coursework 2 additional rules

- You **shall** unit test your code
- You **shall** use GitHub for source code management (throughout development i.e. not simply upload your final code once before submission)
- Instructions how to claim your repo are in the assessment tile on moodle
- You **shall** submit all code and files required to build the app in GitHub classroom
- You **can** use other people's code
  - you can choose to include open source/sample code
  - if so, you **need** to acknowledge every reused bit
  - you need to **respect** the license under which it was provided
- **BUT** you are marked only on your own work
- You **shall** comply with plagiarism regulations cf. moodle for details
- Marks will be awarded for well-commented code

## Deadlines

Coursework 1: 2pm on Friday 10 June

Coursework 2: 2pm on Wednesday 3 July

Full details available on the assessment tile on moodle

Late submission penalties:

- Work that is submitted up to 7 calendar days late will be subject to a late penalty representing a 10% deduction of the original mark awarded
- Any work that is submitted more than 7 calendar days late will receive a late capped penalty mark of 40%
- The absolute cut off deadline for late submission is 2 weeks after which any submitted work will be treated as a non- submission

## Marking Scheme

### Coursework 1

- Full wireframe design of core elements of UI. (30 marks)
- Full wireframe design of extended element of UI. (20 marks)
- Full and detailed use case diagrams. (30 marks)
- Design quality, appropriateness and completeness. (20 marks)

### Coursework 2

- Full implementation of core elements of UI. (30 marks)
- Full implementation of extended element of UI. (10 marks)
- Fully implemented persistence. (10 marks)
- Unit tests providing full coverage implemented. (20 marks)
- Code quality and commenting. (20 marks)
- All coursework submitted to GitHub according to specification and are complete and comprehensive. (10 marks)