COM4510/COM6510 : Software Development for Mobile Devices

Assignment Requirements

This assignment is primarily concerned with applying the ideas that have been/are being presented in the module on methods and technologies for Mobile Computing; please also see the module lectures and labs on Blackboard.

1. Organisation of the Assignment

The assignment is a single assignment for the module worth 100% of the marks; please see the Blackboard site for the exact deadline. The final report submission is electronic via Blackboard and you will also give a brief demonstration.

Workload and Teams

The assignment is intended to account for about 25 hours of each person's work towards the module as a whole, within a team of 4 to 6 (as is typical in industry).

It is important that you make clear what contribution each group member has made in your final report. It is required that each person contributes to each stage of the assignment, but it is up to each group to decide how to divide the work up between individuals. Buddy Check will also be used for Peer Review and to adjust individual marks within your team; typically this will not raise any mark by more than 10%, but may lower marks by more than 10%.

Deadlines

The deadline is absolutely fixed and you should note that group assignment extensions are only given in very exceptional circumstances that affect every member of a group. You should therefore plan your work to aim at finishing your work at least a few days before the deadline – do not leave it until the deadline, just in case anything goes wrong, and you then find that you are late.

Note that the Computer Science department applies **fairly severe penalties** for handing late assignments. Please see detail on the UG and PGT handbooks.

Material Provided

Please see Lecture material, lab class examples and resources as detailed on the lecture notes.

N.B. no third-party code can be used in the assignment, except the code that has been explicitly provided in the lectures. For example, you are allowed to reuse the code given in *the lecture/lab slides. However, you are not allowed to download any code from the Web or to use any other software that will perform a considerable part of the assignment.*

If in doubt, please ask on the assignment forum in Blackboard.

2. Requirements

The **learning objectives** of this assignment are to 'learn by doing' and you will need to show evidence that you have:

- considered the user and usage through design, e.g. by showing hand drawn sketches
- built an app with a flexible sophisticated layout/interface; you should include **Preview**s e.g. in your report
- have used separation of concerns, i.e. including using MVVM with state
- developed local data storage, i.e. on device persistence
- made use of the phone's sensors, e.g. GPS/GeoLocation
- made use of device services, e.g. camera
- worked together as a group

The requirements are to create a Geolocation aware Todo app. You will also need to choose the specific users/focus for the app in agreement with your team; this offers you some choice in the problem you tackle and the solution you develop.

Your solution will need to match the following requirements as well as the above learning objectives.

2.1 The problem

In general the problem is the area of task lists, also known as Todo Lists or Todos. In particular the focus is on Geolocation aware Todos - by recording the location when creating a Todo and also by allowing Todos to become active when a previously chosen location is visited.

You will, as part of your assignment, need to define a set of users and a more specific purpose/usage for their Todos. For example, you might focus on:

- Learner drivers studying for their driving test
- PC owners building themselves their first PC and tracking components, build, etc.
- A student planning and cooking a meal for a number of different guests

I **strongly recommend** that at least one team member is very familiar with your target users/usage; e.g. do not propose 'academics keeping track of conference submission deadlines' or 'racing car driver check list' - since these are users and topics you would have to spend time learning more about.

2.2 Required features

You will design, build and test your Mobile app for recording a single user's Todos. The user must be able to create, update and delete their Todos (tasks).

The app must be built for **mobile Android, using Kotlin and Compose**[1] and should run on Version 11, 12 or 13 (14 is also allowed but not supported). The minimum android version must be version 11 (or earlier).

The app will include image taking/uploading and also be geolocation aware for the purpose of recording location when creating Todos as well as location triggered Todos. For best marks, the app will allow[2]:

1. Entering custom (empty) Todos including a title, description, priority (at least)
2. Taking a picture (and choosing an existing picture) and adding it to a Todo, e.g. a photograph of where I parked my car - such as a picture of the floor number
3. Showing typical tasks (for your users/focus) that a User can copy and fill in further, e.g. finding out if any dinner guests have food allergies or dietary restrictions
4. Subtasks to be held within a Todo, likely as a list of checkboxes, e.g. a checkbox against each dinner guest
5. Recording the (latest known) location on creating a Todo, as well as time and date
6. Including a date/time 'reminder', e.g. '8th Nov at 12.30pm' or 'in 2 days'
7. A notification to be shown when the reminder time is reached
8. **Attaching a location, as latitude and longitude, to a Todo; this should include a 'distance' parameter (circle area)**[3]
9. A notification to be shown when the user arrives at the location of a Todo
10. ~~Notifications should allow a user to easily open the app~~
11. Showing the current Todos ordered by priority, date/time and nearest (location)
12. Users must be able to edit all the parts of a Todo
13. Mark Todos as 'done' and view/recover 'done' Todos
14. Permanent deletion of 'done' Todos with confirmation
15. Your own (extra) requirement - that you will need to highlight in your submitted report. The difficulty and appropriateness of your requirement will be considered when marking.

Note: these requirements also have hidden/implied requirements that you are expected to discover yourself, e.g. you will need to derive how priority is represented, how many values there are, what you do if the user doesn't choose a priority.

You should consider using a software engineering process, e.g. agile, to help you develop as a team, and also to prioritise and identify the key user stories.

[1] Functionality using Java and XML views will typically not be marked
[2] Note: these are deliberately not listed in a specific order
[3] This requirement has changed from:
Picking a (geofenced) location from a map and attaching to a Todo; this should be lat/lon based and will need to include a 'distance' parameter (circle area)

2.3 Demonstration

In week 11 during the labs on friday, you will give a **10 minute demonstration** of your app to show it working. You are **strongly encouraged** to time yourself beforehand. This is an opportunity for you to 'show off' the interactive aspects of your app and you must include your own requirement as part of this. *Note: You should have more than one device ready to demonstrate on, possibly including a virtual device.* You will also be given feedback which you may then use to improve your final submission (report).

You will likely find that some parts of your app will only be visible through your report; also some may be added/updated after the demonstration.

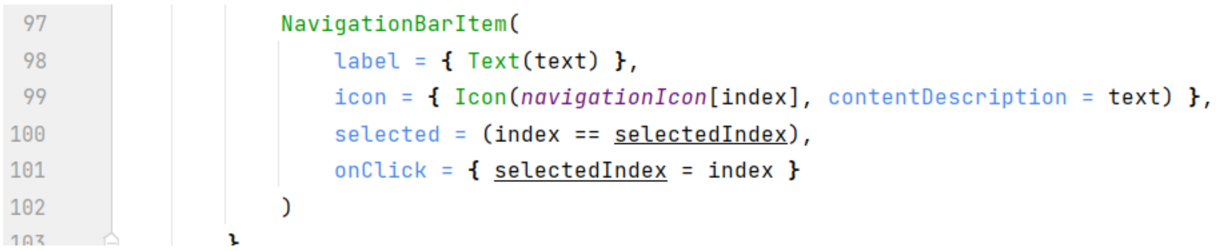
In marking your demonstration (and app), we will be considering:

- Requirement completeness
- Robustness/reliability
- Interface suitability and aesthetics
- How the app copes with portrait/landscape modes and switching to/from the app
- The quality of the solution; a simplistic (functional) solution will attract fewer marks

2.4 Exemplar Code

You will include in your report your exemplar (best) code, limited to a maximum of 4 sides.

This exemplar (kotlin) code allows you to show off your ability and you should choose your code to represent different aspects of your solution. You may include up to 20% of non application, e.g. test code. You must use source code colouring, line numbers and name each listing. Take a screenshot (do not paste as text) and paste it in the document, e.g. as seen in the slides (but with line numbers):



```
 97             NavigationBarItem(
 98                 label = { Text(text) },
 99                 icon = { Icon(navigationIcon[index], contentDescription = text) },
100                 selected = (index == selectedIndex),
101                 onClick = { selectedIndex = index }
102             )
103         }
```

Listing A java/com/example/helloapp/MainActivity.kt

Please note the Caption must include (for best marks):

- an alphabetic listing 'letter', e.g. 'Listing A' above allows feedback to simply refer to A.99 (i.e. listing A, line 99)
- the location from right clicking the file in Android Studio → Copy Path/Reference → Path from Content Root, to allow the marker to see more detail in your repository when necessary
- You should not (typically) include a whole file listing
- You may choose to 'fold' parts of your code
- Standard code quality applies, e.g. appropriate comments, meaningful naming
- Code should, when possible, explain itself and not need many comments, e.g. use meaningful function/variables names

I strongly recommend you do not paste code directly into your document – it frequently creates issues when opened for marking – a screenshot is quicker and more reliable.

2.5 Code Committed to the Team Repository

We will also be looking at the last commit (i.e. at the deadline time) to the team's Git repository for marking. We may run code within a virtual device or a Nokia phone, typically where a demonstration has left us uncertain about the quality of the app, or inspecting code where we have concerns about the report being realistic. In general, we will prefer to base marking on the demonstration and the report.

N.B. Do not send files after the deadline that you forgot to commit.

Please note the following important points in regard to the use of the Git repository:

1. This must be the Team repository (Gitlab) and no other will be accepted. Ensure you commit to your team repository — not your personal repository or another Git repository
2. All members should be making regular commits. In marking your work, we will be checking that you have been making regular commits — not just one or two commits or a sudden rush of commits just before the deadline
3. You should attempt a clean install/run before your final commit

I strongly suggest your team organises itself with a 'soft' deadline 1–2 hours (or more) before the hard deadline. After your soft deadline, make final checks to ensure everything is committed and runs properly. If you've forgotten anything, you'll still have time to correct the mistake. Don't use the period after the soft deadline to make any code changes.

3. Marking schema

The marks for the team will be calculated as follows

- Demonstration - 20%
- Report - 80%

Your report will be marked according to the following sections; where the marks shown add up to 80% for the report. These should be in the order given to reduce the probability that your work is missing or missed:

- Section 1 - Requirement - 10%
  - including your chosen users/focus, your additional chosen requirement, and any additional assumptions you have made to the requirement
- Section 2 - UI Design - 15%
  - including design sketches (you may wish to put some in an appendix to show the process but not directly be marked). These should be low fidelity, hand drawn, sketches. You should also show the navigation structure of the app.
- Section 3 - Implementation - 20%
  - this will include the exemplar code. You should also explain some of the choices

you have made as well as why you made those choices. Your code will also be inspected to help determine this mark

- Section 4 - App Brochure - 15%
  - here you will show examples of the use of your app using screenshots and descriptions. You should not directly repeat your demonstration, but should show different examples and interactions, as well as highlighting where you have made changes from demonstration feedback, as well as areas you did not have time to show
- Section 5 - Reflection - 10%
  - here you will reflect on what you have learnt, identifying where you feel there may be good opportunities to improve the fit of your app for your users/focus. You should also include a proposal for the next task you would perform
- Section 6 - Work allocation - 10%
  - here you will need to show how you planned your work and the split between your team members, e.g. task allocation, likely including the person responsible for a task as well as the persons who actually worked on the task. N.B. the best plans will show how each team member was allocated and performed specific tasks within each of the other sections/requirements

3.1 Assessment Criteria

Note: All requirements for a lower level must be satisfied before a higher level mark may be given.

The table below will be used when marking your report and the demonstration:

Grade	Description	Weighted Marks		
		10%	15%	20%
Hard Fail 0-39%	Few (or none) of the features have been implemented and/or the quality is very low. Little regard is shown for the users/usage. Design is either missing, very shallow or insufficient, e.g. only covers a few of the requirements. The implementation is missing or unreliable or too incomplete to be acceptable. Evidence of app usage (brochure) is missing or incomplete. Reflection is missing or too shallow. Evidence of teamwork, and planned allocation of work, is missing or very weak. The Git repository shows a minority of the team submitting work and/or submission on the last day.	0-3	0-6	0-7
Fail (soft) 40-49%	Some, typically a half, of the features have been implemented to some acceptable degree. There is more work to be done to make the quality acceptable. Some useful consideration of the users/usage is evident. Design is partly acceptable but incomplete and/or low quality. The implementation shows some useful learning but the breadth or quality are not acceptable. Evidence of app usage (brochure) exists but is not yet sufficient.	4	7	8-9

	Reflection has some value but is either shallow or too brief. There is some evidence of teamwork, though the planned allocation is poor with team members likely working only in specific areas. The Git repository shows late and/or very uneven work submission; typically most submissions are in the last week.			
Pass 50-59%	Most of the features have been implemented, though the quality/fit may be weak for a few. Consideration of the users/usage is evident and appropriate for most of the features. The evidence of design and shown design is acceptable, though likely to be lacking in consistency, quality and/or appropriateness. The implementation is good though it may fall down slightly in reliability, quality, consistency. At least one of services/sensors[4] must be working to a degree. Evidence of app usage (brochure) is good and covers most of the expected interactions. Reflection is acceptable though typically lacking in depth or impartiality. There is evidence of teamwork, with a fair allocation of work, though team members do not contribute to all the areas. The Git repository shows an uneven submission of work, though submissions are across most of the expected development weeks (7 to 11).	5	8	10-11
Merit 60-69%	Almost all of the features are implemented, though a very few areas might be missing or low quality. The users/usage have been identified and there is a good user fit. The design is mostly at least good, with only a very few inappropriate design choices. The implementation is good and with a little polish would be useful. Both services/sensors are mostly working. Evidence of app usage (brochure) is very good and covers almost all of the required interactions. Reflection is mostly impartial and generally good with some relevant insights. Team work allocation is well planned and fairly allocated, likely lacking actual evidence. Git submissions support a team working well together.	6	9-10	12-13
Distinction 70-100%	All of the features have been implemented. The users/usage are very well considered and the fit is very good. Evidence of design is complete (or very nearly complete) with an excellent design shown and very good design choices shown. The implementation is excellent and could be deployed and used. Both services/sensors are fully (or very nearly fully) working. Evidence of app usage (brochure) is excellent and covers all of the required interactions. Reflection is impartial and very good with some insightful observations likely linked outside the module content. Team work allocation is very well planned and allocated with good evidence shown including within the Git repository.	7+	11+	14+

[4] e.g. camera service, Geolocation sensor

3.2 Group size

Teams of 5 and 6 are expected to produce the same amount of work, due to inefficiencies in communication, etc. Teams of 4 are still expected to produce all the same requirements, but an allowance will be made for quality of the requirements compared to larger teams.

If any teams have members leave and become a team of 3, please let me, Andrew Stratton, know asap by email that this has happened.

3.3 Anti-cheat measures

Please note that measures are in place for detecting plagiarism and in general cheating.

4 Queries about this assignment?

Should you have any queries about the assignment, feel free to post on the Blackboard forum or contact the module leader a.strattton@sheffield.ac.uk