COMP1424 (2019/20)	**Mobile Application Development**	**Faculty Header ID: 300837**	**Contribution: 80% of course**
Course Leader: Dr Markus Wolf	**Practical coursework**		**Deadline Date: Monday 02/12/2019**
This coursework should take an average student who is up-to-date with tutorial work approximately 40 hours Feedback and grades are normally made available within 15 working days of the coursework deadline			
Learning Outcomes: A, B, C			

Coursework Submission Requirements

- **An electronic copy of your work for this coursework must be fully uploaded on the Deadline Date of** Monday 02/12/2019 **using the link on the coursework Moodle page for COMP1424.**
- **For this coursework you must submit a single PDF document. In general, any text in the document must not be an image (i.e. must not be scanned) and would normally be generated from other documents (e.g. MS Office using "Save As .. PDF"). An exception to this is hand written mathematical notation, but when scanning do ensure the file size is not excessive.**
- **For this coursework you must also upload a single** ZIP **file containing supporting evidence.**
- **There are limits on the file size (see the relevant course Moodle page).**
- Make sure that any files you upload are virus-free and not protected by a password or corrupted otherwise they will be treated as null submissions.
- Your work will not be printed in colour. Please ensure that any pages with colour are acceptable when printed in Black and White.
- **You must NOT submit a paper copy of this coursework.**
- **All courseworks must be submitted as above. Under no circumstances can they be accepted by academic staff**

The University website has details of the current Coursework Regulations, including details of penalties for late submission, procedures for Extenuating Circumstances, and penalties for Assessment Offences.
See http://www2.gre.ac.uk/current-students/regs

Detailed Specification

Please read the entire coursework specification before starting work.

You are to implement a mobile app which is a prototype of a mobile app for use by people reporting on sporting events such as football matches. It is called the Mobile Sports Event Reporter (**MOBSER**). It should support the features listed
below:

- Features **a)** to **e)** are to be implemented as a **native Android app coded in Java**.
- Feature **f)** is to be implemented as a **hybrid app coded using PhoneGap/Cordova**.
- Feature **g)** can be implemented as **either or both** additions to the native android app or PhoneGap app.

Your final app is the culmination of all your hard work on this course, which should become a strong addition to your programming portfolio. You should produce an app that is well-designed, robust and useful. The GUI design should be clean, simple to navigate, and operate smoothly without sluggishness or crashes. The app should not require instructions or a manual to use.

1. Description of the application

a) Enter details of events being reviewed (10%):

Note that users must be able to enter all of the following fields. ***Required*** fields mean that the user must enter something in this field; otherwise they will be prompted with an error message. ***Optional*** fields mean that the user can enter something if they wish, but they will not get an error message if nothing is entered.

The user/critic should be able to enter:

- Date of the event - *required field.*
- Start time of the event - *required field.*
- The name of the sport. This should be selectable from a list of at least three sports. You decide what sports you will allow (e.g. "football", "ice hockey", and "basketball"). *Required field.*
- The names of the participants (e.g for a football match this would be the names of the teams). - *Required field.*
- Location of the event. *Optional field (i.e. the user may or may not enter it)*
- Email address of the person reporting the event. *Optional field.*
- Two or more other fields of your own invention – be creative!

The app will check the input and if the user doesn't enter anything in one of the required fields, the app should display an error message.

Once the details have been accepted by the app (e.g.. no required fields were missing), it should display the details back to the user for confirmation and allow them to go back and change any details that they wish.

b) Store, view and delete event details or reset the database (10%)

All the details entered by the user should initially be stored on the device in an SQLite database.

The user should be able to **list all** the **details of all events** that have been entered into the app, edit or delete individual events, and delete all the details from the database.

c) Enter the score as the sporting event progresses (10%)

The user should be able to enter details of the score as the event is in progress. How this works will depend to some extent on the sports you choose. For instance, if the event is a football match then the user should be able to make an entry whenever a team scores a goal.

The score should be stored on the device in the SQLite database.

d) Search (10%)

The user should be able to search for an event in the database by name. At its simplest, this could mean entering the name and displaying the first event that matches. Ideally the user should be able to enter the first few letters of the name and display all matching events.

Advanced search options will allow searching for all events with the following criteria: a particular area, event type or date range. It should be possible for a user to select an item from the resulting search list and to display its full details.

e) Upload details to a cloud-based web service (10%)

The user should be able to upload all the basic details of events stored on the mobile device to a cloud-based web service. It should be possible for all events to be uploaded at once. YOU DO NOT NEED TO WRITE the web application as it will be made available to you. See the Moodle site for this course in week 7 for the URL of the web service and further information about how to use it (http://gillwindallapp1.appspot.com/madservlet).

The web service will accept the data and reply with a message which your app should display. Ideally your app should format the message so that is easy for the user to read.

The format in which the data must be uploaded

The data you upload to the web service should be sent in an HTTP POST message. The HTTP POST message will contain one field called **jsonpayload**. The value of **jsonpayload** is a string in JSON format.

You have some flexibility over how you design the JSON however the following are essential:

- The JSON must be valid. See http://www.w3schools.com/js/js_json_syntax.asp for details on JSON syntax.
- You must include a field with the name "userId" which has a value of your university user id eg. "userId":"wg999"
- The event details must be represented as an array with the name "detailList"
- Each element in the array "detailList" represents an event about which details are being uploaded.
- Each event in "detailList" must at a minimum have a field called "name" whose value is the actual event’s name or the brief description of the event.
- Below is an incomplete example of what the JSON might look like
  {"userId":"wg999",
  "detailList":[
  {"name":"Arsenal Manchester match", *other fields for event*},
  {"name":"European Basketball final", *other fields for event*}]}

The format of the response

The web service will respond with an HTTP response containing JSON which represents a reply to your upload. The JSON reply will contain the following fields:

field name	value
uploadResponseCode	A code indicating whether or not your upload was successful. Values are either SUCCESS or FAILURE. If the code is FAILURE the message field will contain more information about what went wrong.
userid	The same userId that was included in the upload message
number	The number of events uploaded
names	The names of the events uploaded
message	A message giving more information about what happened. Particularly important if the uploadResponseCode is not SUCCESS

An example response is given below. Note that the order of the fields may vary

```
{"uploadResponseCode":"SUCCESS",
"userid":"wg999",
"number":2,
"names":"Arsenal Machester match, European Basketball final",
"message":"successful upload – all done!"}
```

Your app should display the data sent in the response. You could just display the raw JSON message, but it would be better to format the output to make it more readable for the user.

f) Create a cross-platform prototype of the app using PhoneGap (10%)

Implement as much as you can of features a) and b) using PhoneGap.

g) Add additional features to either or both the Android or PhoneGap version of the app (10%)

Features a) to e) are the core requirements for the app. If you have implemented these and want to add some additional features, then you may. Any enhancements should be implemented **in addition to NOT instead of** the core requirements. The idea is that these features **stretch** your skills, so be prepared to do your own research and feel free to show off! You can think of your own enhancements. Here are some possible examples:

- Allow photos taken by the camera to be added to the data stored
- Pick up the location automatically from the user's location
- Show the location of an event on a map
- Make use of an external web service other than the one provided for the coursework.
- Anything you can think of – again be creative!

2. Report (30%)

Write a report consisting of **all** the following sections:

- **Section 1. (5%)** A **concise table** containing a checklist of the features you have been able to implement. Please refer to the features list given above in the specification. For example, you might write:

Feature	Implementation
a)	Fully implemented
b)	Fully implemented
c)	I have created the user interface for data entry but the data is not being stored ☹
d)	Implemented but the app throws an exception if no matching result is found.
e)	Fully implemented
f)	I have create a PhoneGap prototype of feature a) but not feature b)
g)	No additional features implemented

- **Section 2. (5%(** A concise list of any bugs and/or weaknesses in your app(s). It is unlikely that there are no bugs or weaknesses but if you don't think there are any then say so. Bugs that are declared in this list will lose you fewer marks than ones that you don't declare!

- **Section 3. (5%)** A brief (less than half a page) description of any special strengths of your app(s) that you think should be taken into account in awarding a mark. Please be very specific and realistic in this: vague statements such as "it is easy to use" or "it is well designed" will not gain you marks.

- **Section 4. (5%)** Screen shots demonstrating each of the features that you have implemented. Give captions or annotations to explain which features are being demonstrated.

- **Section 5. (10%)** An evaluation of your app(s). Write between 700 to 1000 words evaluating the app(s) that you have produced**.** Be specific and justify any statements you make. Just saying things like "my app is well designed" without justifying the statement will not gain you any marks. Also, explain how your app could be improved. Again, you need to try to be specific e.g. saying something like "It needs to be made more secure by adding security features" will not gain marks. Your evaluation should include, but need not be limited to, the following aspects of your app:

  i. Human computer interaction (you will have a lecture about this)
  ii. Security (you will need to research this yourself)
  iii. Ability of the app to run on a range of screen sizes and how this could be improved (we will touch on this in the course but you will need to do additional research)
  iv. Changes that would need to be made in order for the app(s) to be deployed for live use

  This sort of discussion will form an important part of your MSc project report so use this opportunity as a way of practicing your skills in writing an evaluation.

Deliverables

1. A zip file containing all the files required to run your app(s). Please try to structure your work so that it is easy for the person marking your work to compile and run your app(s) if they need to. Any compilation, installation or running instructions should be included in a "readme" file.

   If you have **borrowed code or ideas** from anywhere other than the lecture notes and tutorial examples (eg. from a book, somewhere on the web or another student) then include a reference showing where the code or ideas came from and comment your code very carefully to show which bits are yours and which bits are borrowed. This will protect you against accusations of plagiarism. Be aware that **the marker will look for similarities between your code and that submitted by other students** so please do not share your code with any other students as this is considered to be plagiarism. Note that **the upload of this zip** file **is MANDATORY**. It must be **uploaded by 08/12/18** along with your report (deliverable 3) **or you will lose marks and are likely to fail the coursework**.

2. An Acceptance Test of your app and what you have achieved. This will be approximately 15 minutes, and you will be asked to show your app(s) running on a machine in the labs at Greenwich. You will be asked questions about your app(s) and be expected to show an understanding of the code that you have written and the design decisions that you have made. For instance, the tutor may ask you to talk them through the code that implements a particular feature. If you are unable to answer questions about your code you may be investigated for plagiarism. If you develop your app(s) at home it is **your responsibility** to make sure that it runs in the labs at Greenwich. If it will not run on the labs machine you will be allowed to demonstrate it on your laptop **BUT you may lose marks**. Please allow yourself plenty of time to get your app(s) working on the lab machines before your Acceptance Test time. The date, time and place of the Acceptance Test will be announced on the Moodle site for the course.

3. A report consisting of **all** the sections described in section 1.2 of the detailed specification. This is the **final** deliverable.

Grading Criteria

This coursework will not be marked anonymously.

Each of the 7 features are worth 10% of the overall assessment. This doesn't meant that you will automatically obtain the full 10% for a feature for implementing it, but it will be graded based on the assessment criteria below. Section 5 of the report is also worth 10% and each remaining section of the report is graded out of a maximum of 5%.

For a very high distinction (90% to 100%)

- A native Java android application fully implementing all features. No bugs and negligible weaknesses. Exemplary quality code.
- Two or more creative additional features (g) to both apps.
- Acceptance Test: able to show good knowledge of code implemented at the Acceptance Test. Not only able to answer questions about **how** you did something but also **why** you chose to do it in a particular way and **compare** with possible alternative implementations.
- Report complete, accurate and easy to read. Section 5 within specified word count, logically structured and making some insightful points about **all four** of the issues specified.

For a high distinction (80% to 89%)

- A native Java android application fully implementing at least features a), b), c), d) and e). No bugs and very minor weaknesses. Outstanding quality code.
- A working hybrid prototype (f) and one or more creative additional features (g) to both apps.
- Acceptance Test: able to show good knowledge of code implemented. Not only able to answer questions about **how** you did something but also **why** you chose to do it in a particular way.
- Report complete, accurate and easy to read. Section 5 within specified word count, logically structured and making some insightful points about **all four** of the issues specified.

For a distinction (70% to 79%) the following is required

- A native Java android application fully implementing at least features a), b), c), d) and e). Very few minor bugs or weaknesses. Excellent quality code.
- A working hybrid prototype (f) and one or more very good additional features (g) to one or both apps.
- Acceptance Test: able to show good knowledge of code implemented. Not only able to answer questions about **how** you did something but also **why** you chose to do it in a particular way.
- Report complete, accurate and easy to read. Section 5 within specified word count, logically structured and making some insightful points about at least three of the four issues specified.

For a merit mark in the range 60 to 69% the following are required:

- A native Java android application fully implementing at least features a), b), c) and a good attempt at d) and e). Few minor bugs or some weaknesses. Very good quality code.
- A working hybrid prototype (e).
- Acceptance Test: able to show good knowledge of code implemented.
- Report complete, accurate and easy to read. Section 5 within specified word count logically structured and making sensible points about at least two of the four issues specified.

For a pass mark in the range 50 to 59% the following are required:

- A native Java android application fully implementing at least features a) and b) with an attempt at c) and d), possibly with some bugs and weaknesses. Good quality code.
- Hybrid prototype (e) attempted.
- Acceptance Test: able to show good knowledge of code implemented.

- Report complete including some attempt at section 5

For a marginal fail mark in the range 40 to 49% the following are required:

- A native Java android application with a reasonable attempt at feature a) and some attempt at b) and c).
- Acceptance Test: able to show reasonable knowledge of the attempted features.
- Report mostly complete.

For a fail mark in the range below 40%:

- A native or hybrid android application attempting feature a), b) and c), but buggy.
- Acceptance Test: unable to show reasonable knowledge or understanding of the attempted features.
- Report missing or mostly incomplete.

For a coursework that does not fit into any of these categories (e.g. features a) to d) are implemented but section 5 of the report is less than 700 words and weak) the grade will be determined by taking the lower mark applicable (eg. range 50% in this example) and making some upward adjustment for the other aspects of the coursework. Please be aware to pass you must submit a working app **and** an acceptable report. For instance even if all features a) to e) are implemented but the report is very, very weak or has several sections missing then you may fail the coursework.

NOTE: Failure to do your Acceptance Test will normally result in you being awarded 0% for the coursework. Even if your implementation and report are excellent and would be awarded a mark of 80% but you don't do an Acceptance Test then you may score 0% for the coursework.

Assessment Criteria

Your app(s) will be assessed on the following criteria.

- **Features implemented.** The number of features (listed as a) to g) in the specification above) that you have successfully implemented will have a big effect on your overall mark.

- **The quality of the application code you produce**. Credit will be given for inclusion of meaningful comments in the code, use of the sensible naming standards (eg. for packages, classes, variables, and methods), code layout (eg. indentation to make the structure of "if" statements and loops clear), avoidance of unnecessary duplicate code, for the Android app use of appropriate Java language features and Android APIs. Java coding conventions (covering naming and indentation etc) can be found at http://www.oracle.com/technetwork/java/codeconvtoc-136057.html – (please note that this Oracle page is no longer maintained as the java conventions are common standard, but the page continues to be used as a sources of reference).

- **The user interface.** This is not a course about user interface design but credit will be given for making your application as pleasant an experience as possible for the user. Examples of good practice are: allowing the user to choose options rather than their having to type in input, sensible default values, validation of input, and meaningful messages. Credit will be given for showing the use of a range of appropriate features from the Android GUI API.

Your report will be assessed on the following criteria.

- Are all the required sections included and completed properly?

- Does the report give an accurate reflection of what you have achieved?

- Is the report clear and easy read? Does it follow the structure specified?

- Is the evaluation (section 5) realistic and does it show that you have really thought about your app(s) and the specified issues as well as how they may be enhanced to be ready for live deployment. Do you show insight into the complexities of app development and the challenges of balancing the various constraints involved?