MANCHESTER METROPOLITAN UNIVERSITY
School of Computing, Mathematics & Digital Technology

ASSIGNMENT COVER SHEET

Unit:	6G5Z1001 Advanced Programming
Assignment set by:	Dr Mohammed Kaleem
Verified by:	Dr Nick Whittaker
Moderated by:	Dr Nick Whittaker
Assignment number:	2CWK50
Assignment title:	Android Application Development
Type: (GROUP/INDIVIDUAL)	Individual
Hand-in format and mechanism:	Submission is online, via Moodle. More information is available in the attached coursework specification.
Deadline:	As indicated on Moodle.

Learning Outcomes Assessed:

1) Design and implement internet connected mobile applications.
2) Apply mobile device programming techniques to web server interactions.

It is your responsibility to ensure that your work is complete and available for assessment by the date given on Moodle. If submitting via Moodle, you are advised to check your work after upload; and that all content is accessible. Do not alter after the deadline. You should make at least one full backup copy of your work. Penalties for late hand-in: see Regulations for Undergraduate Programmes of Study:

http://www.mmu.ac.uk/academic/casqe/regulations/assessment.php.

The timeliness of submissions is strictly monitored and enforced.

Exceptional Factors affecting your performance: see Regulations for Undergraduate Programmes of Study: http://www.mmu.ac.uk/academic/casqe/regulations/assessment/docs/ug-regs.pdf

Plagiarism: Plagiarism is the unacknowledged representation of another person's work, or use of their ideas, as one's own. MMU takes care to detect plagiarism, employs plagiarism detection software, and imposes severe penalties, as outlined in the Student Handbook http://www.mmu.ac.uk/academic/casqe/regulations/docs/policies_regulations.pdf and Regulations for Undergraduate Programmes (http://www.mmu.ac.uk/academic/casqe/regulations/assessment.php).
Bad referencing or submitting the wrong assignment may still be treated as plagiarism. If in doubt, seek advice from your tutor.

Assessment Criteria:	Indicated in the attached assignment specification.
Formative Feedback:	Formative feedback will be provided in laboratory sessions with an assignment check point
Summative Feedback format:	Grid can be found at the end of this specification. Students will receive written feedback with their final marks.
Weighting:	This Assignment is weighted at 50% of the total unit assessment.

Advanced Programming – Assessment 2

Android Application Development

Introduction

Deadline: 19th March 2018

For this assignment you will utilise the Java HTTP Server that you implemented in the first assignment and develop an android application using android studio that will (at the minimum level) communicate with the server in order to retrieve the student data in JSON format, parse the data appropriately and display it on screen.

You are being asked to submit **two deliverables**. **Firstly**, you should submit a complete zipped copy of your **source code** (i.e. the android app source code). **Secondly**, you will need to produce a "**screencast**" showing you running your completed app and interacting with it in the android emulator. More details on how to produce this are included later in this specification. You must submit **both the zipped source code and the screencast video** before the deadline.

Learning Outcomes

Design and implement mobile applications. Apply mobile device programming techniques to web server interactions.

The specification for the android app is presented below. Each section has been assigned a weighting to give you an idea of the relative importance of each tranche of functionality in the marking scheme. You will be marked on how much of the specification you have implemented overall, as evidenced by review of your submitted source code, and by viewing your submitted screencast.

Basic App Functionality (40%)

You should complete this part of the coursework using the **android studio** development environment, as is installed on the lab computers. You are free to use a physical android device, with hardware debugging enabled to debug your application if you wish, but marking will be conducted in the android emulator (i.e. your screencast must be of the emulator) so I would recommend using that instead. The emulator should be configured to emulate a Google Nexus phone, running API 21, Android 5.0 (Lollipop).

The app you develop will be similar to the contacts app we all have in our smart phones today as illustrated in figure 2, except your app will display student information. The scenario is that you are developing this app for the head of school, who needs to have up to date student information to hand at all times.

With this in mind your app should at the very least **connect to the RESTful student web service** that you developed in assignment one (**or use the Student REST API available online** (more details on this below)) in order to **retrieve the student data in JSON format**. The app should then **parse the JSON data** and display it on screen in an appropriate way (i.e. ListView). **In addition**, when the user clicks on a student in the list **another activity** should be presented showing **ALL** that **student's information**.

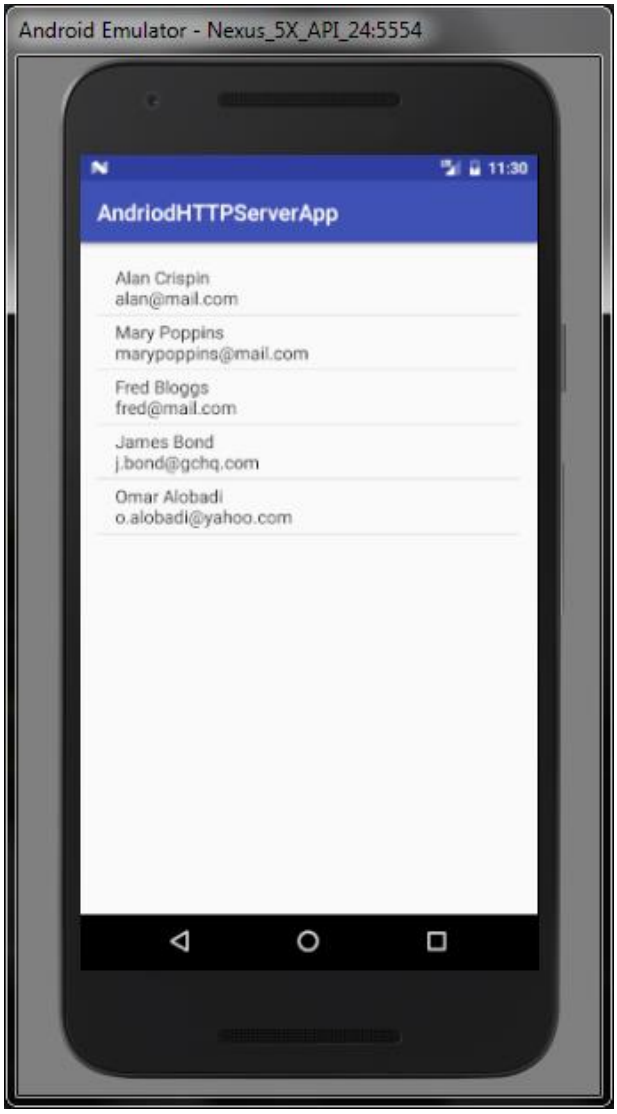

Figure 1 Android App

*Additional functionality can be added to the app for extra marks, see below for details.

Code Quality & Screencast (10%)

Your **code should be** sensibly **formatted**, laid out **consistently throughout** and well **commented**, accompanied by a screencast demonstrating your app's functionality. All the features implemented in the app should be clearly demonstrated in the screencast. Further information regarding the screencast can be found below.

Additional Functionality

These tasks are to be completed with some additional research on your part. They are quite challenging therefore you will be expected to put in some considerable effort to achieve these marks.

Threading (AsyncTask) (20%)
The tasks of talking to the server to retrieve data can sometimes take some time to complete. Leaving your app's interface locked up whilst it performs this task is hardly going to enhance the user experience! In addition to this, making network calls on the main app thread is bad practice in the app development world.

Improve matters by handling the network calls that retrieve the data from your Java server (or the provided Student REST API) using the AsyncTask class, which will perform them in a **new thread in the background**. *Remember to remove the StrictMode workaround discussed in lectures once threading is working.*

Add Student to the Server Database from the App (10%)
Implement a way for the user of your app to be able to **add new students to the server's database**. This will typically involve creating a **new Activity** in the app that **displays a form** that can be filled in by the user with a **button to submit the form** data to the server. The data entered in to the form by the user will need to be processed and sent to the RESTful server through the appropriate HTTP method (i.e. POST).

Update Existing Student Details (10%)
Implement a way for the user of your app to be able to **update existing student details on the server database**. There are many ways this could be implemented. The simplest method would be an Activity that is launched from the student details screen that displays editable text fields (i.e. EditTexts) filled out with the existing user details which the user can then amend/update and submit to the server which will update that specific student record in the database.

Delete Student (10%)
Implement a way for the user of your app to be able to **delete a student's details/record on the server database** (a long press on the ListView item with a popup confirmation message would be a good idea here).

The Screencast

The android emulator is slow. Furthermore, it requires a reboot between marking each student's work. As such, it would simply be infeasible for us to run every submission in the emulator, and get your marks back to you in a timescale that you would be happy with. To ameliorate this, you shall make a screencast of you performing some set tests on your android app. We'll be viewing the submitted screencasts using the MPlayer video player (see here). This should be able to play almost all formats generated by popular tools. However, I'd recommend the **Open Broadcaster Software** (see here) as being capable of making full-screen videos, including the android emulator. In the mac labs, this service requires the Java security level to be reduced to "medium", so you may wish to use the installed **QuickTime screen recording** tool instead. Your recorded screencast must demonstrate all of the functionality you wish to be marked on, performing the tests set in the released testing document, and be **no more than two minutes in length.** Moodle also places an upper limit on the size of an assignment upload, which is 100Mb. You are welcome to use a microphone and talk us through the submission if you wish, but this is not compulsory. Lastly, you will **lose marks** if you fail to submit a screencast.

The start (first 5 seconds) of the screencast **MUST** show the screencast submission template filled out with the details of your submission. The template will be made available on Moodle closer to the submission date.

The Student RESTful API

For those of you who didn't finish the first assignment a RESTful API has been set for you to use for this assignment. The service is available at: http://radikaldesign.co.uk/sandbox/studentapi. You can create a new account and then use RESTful routes to create, retrieve, update and delete student information in a similar way to the pervious assignment. In order to use any aspect of the service you will need an API key. Your key can easily be retrieved once you've created an account. The website also allows you to easily enter some dummy/test student information through a web form. Full documentation on how to use the service is available on the website on the "Api docs" page.

How You Will be Marked

Your tutor will be reviewing the **code** and the **video** for each submission. You will receive a mark for each of the tranches of functionality out of the maximum mark indicated in each section heading. These will be added together to give your final grade for the assessment. The maximum mark indicated in each section heading is the mark allocated to a solution that fulfils that part of the assignment brief perfectly.

Mark Scheme

Your work will be graded in a number of areas, attracting a number of marks for each. Guidance on the assessment criteria for each area is included below. For each area you will be given a mark and these marks will then be combined to give an overall mark for 2CWK50, which is worth 50% of the final unit mark.

Basic App Functionality 40%		**Code Quality & Screencast 10%**		**Additional Functionality Tasks**							
				AsyncTask (20%)		**Add New Student (10%)**		**Update Existing Student (10%)**		**Delete Student (10%)**	
No meaningful attempt.	0 marks	No meaningful consideration to code quality and app design /Layout.	0 marks	No meaningful attempt at task.	0 marks	No meaningful attempt at task.	0 marks	No meaningful attempt at task.	0 marks	No meaningful attempt at task.	0 marks
App successfully connects to the server in order to retrieve and display the student information on screen *(ListView with name and email).*	10–20 marks	Code contains minimal comments.	3–5 marks	Some attempt made but not working correctly.	1 - 5 marks	Some attempt made but not working correctly.	1-5	Some attempt made.	1-5 marks	Some attempt made.	1-5 marks
All the above with a details activity where ALL student information is displayed (name gender, DOB, address etc etc).	40 marks	All code submitted is well structured, laid out consistently and commented. Good use of functions to stop code repetition wherever possible. WITH a screencast demonstrating app functionality following the example on Moodle.	10 marks	AsyncTask successfully implemented. **ALL** network calls are performed in an AsyncTask.	20 marks	App has a functioning mechanism for adding students to server database (i.e. an activity with a form and submit button which posts to the server).	10 marks	App has a functioning mechanism for updating students information on the server database (i.e. an activity with a form and submit button which posts to the server updated student details).	10 marks	App has a functioning mechanism for deleting students from the database (ideally, long click on a list item with a confirmation message).	10 marks

If you need any help in understanding this assignment please talk to the tutor who takes you for your laboratory sessions or arrange to see either Dr Alan Crispin or Dr Mohammed Kaleem during their office hours.