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

Vehicle DB REST API Server and Android Application Development

Introduction

Deadline: See Moodle

This piece of coursework is split in to two distinct parts. For the **first part** you will develop a web server in Java (HTTP or Jetty) which provides access to vehicle information stored on an SQLite database through a RESTful web service. Upon receiving a GET request the RESTful Java server will output data in JavaScript Object Notation (JSON) format. The server will also support POST, UPDATE, DELETE requests in order to allow inserting new vehicle records or updating and deleting existing vehicle records.

The **second part** you are tasked to develop an android application using android studio that will communicate with the server in order to retrieve the vehicle data in JSON format, parse the data appropriately and display it on screen. As well as allowing the user to insert, update and delete vehicle data through the app.

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

Learning Outcomes

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

The specification for the both the server and 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.

Your Server/Web Service (20%)

You will implement your own **RESTful web service in Java** using **HTTP Server** or **JETTY** which will accept client GET requests and **return data in JSON format**. The server will connect to a SQLite database that contains vehicle information (similar to the database from the first assignment) and return that data to the requesting clients. The default URL (http://localhost:8005/api) should output all the vehicle information stored in the database.

JSON Format of Returned Data
The data returned by your server will be in JSON format. JSON stands for **JavaScript Object Notation**, which is a lightweight data-interchange format. JSON is quickly becoming an industry standard method of data exchange over existing HTTP protocols.

JSON format is an array of individual JavaScript objects containing key value pairs. The data your server will return should resemble the following example:

```
[{"vehicle_id":1,"make":"Ford","model":"Fiesta","year":2014,"price":9500,"license_number":"AB14CDE","co
lour":"Silver","number_doors":5,"transmission":"Manual","mileage":20000,"fuel_type":"Petrol","engine_size
":999,"body_style":"Hatchback","condition":"Good","notes":"Eco Boost"},
{"vehicle_id":2,"make":"Toyota","model":"Yaris","year":2015,"price":9000,"license_number":"AB15CDE","c
olour":"White","number_doors":5,"transmission":"Manual","mileage":14000,"fuel_type":"Petrol","engine_siz
e":1300,"body_style":"Hatchback","condition":"Good","notes":"Icon"}]
```

****Additional functionality can be added to the server for extra marks. See additional functionality section for details.***

Your App (20%)

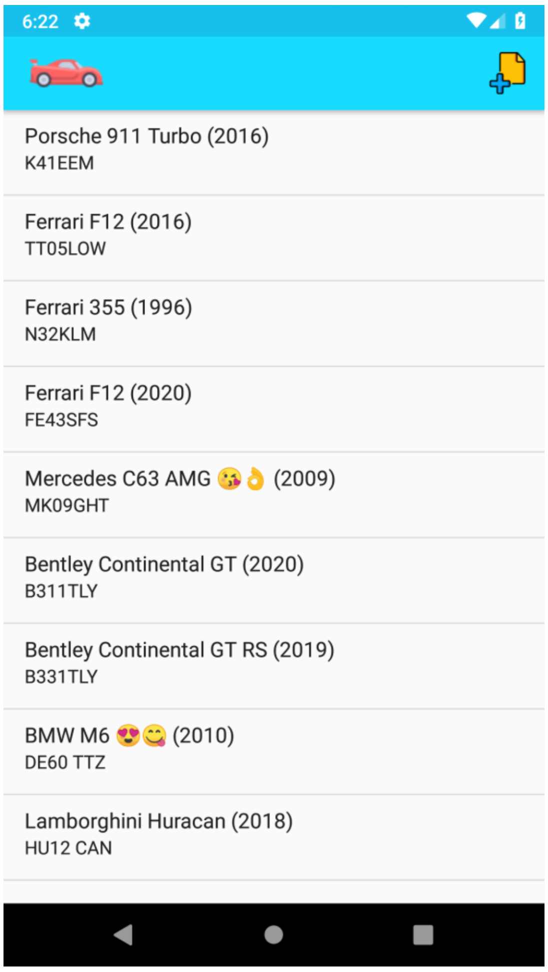
You should complete this part of the coursework using the android studio development environment, as is installed on the lab computers. You are free to use a physical android device, with hardware debugging enabled to debug your application if you wish, but marking will be conducted in the android emulator (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 Pixel phone, running API 21, Android 5.0 (Lollipop).

The app you develop will be like the contacts app we all have in our smart phones today as illustrated in figure 2, except your app will display vehicle information. The scenario is that you are developing this app for the manager of a used car sales showroom. The manager wants an app that the employees can use anywhere within the vast showroom in order to answer customer queries related to the cars they have in stock.

With this in mind, your app should at the very least **connect to the RESTful** web service server that you created in Java in order to **retrieve the vehicle data**. The app should then **parse the JSON data** appropriately and display it on screen in an appropriate way (i.e. ListView). **(10%)**
For the **full 20%** the app should **implement a details activity** where full details of a vehicle can be viewed once the user clicks/taps on a vehicle from the ListView.

****Additional functionality can be added to the app for extra marks. See additional functionality section for details.***

Main Activity



6:22

Porsche 911 Turbo (2016)
K41EEM

Ferrari F12 (2016)
TT05LOW

Ferrari 355 (1996)
N32KLM

Ferrari F12 (2020)
FE43SFS

Mercedes C63 AMG 😘👌 (2009)
MK09GHT

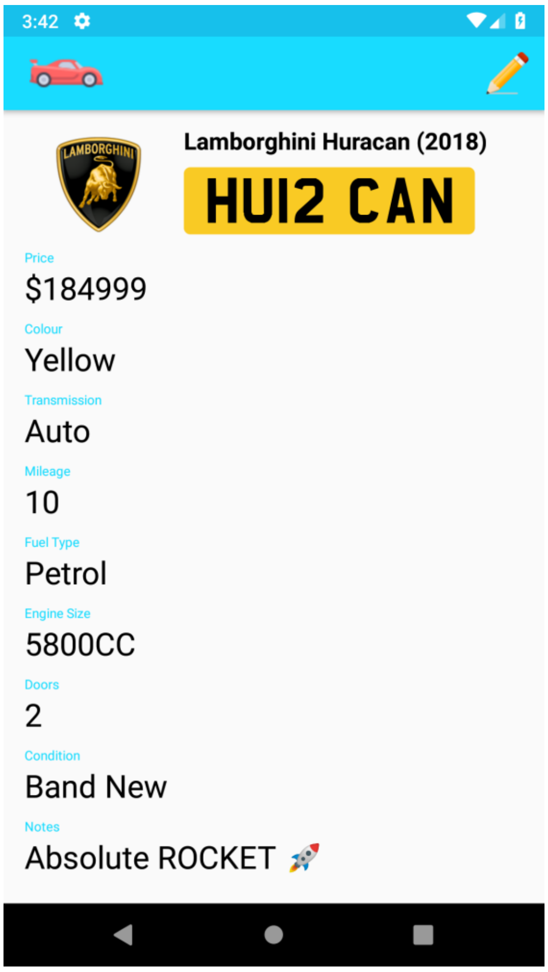
Bentley Continental GT (2020)
B311TLY

Bentley Continental GT RS (2019)
B331TLY

BMW M6 😍😋 (2010)
DE60 TTZ

Lamborghini Huracan (2018)
HU12 CAN

Details Activity



3:42

Lamborghini Huracan (2018)

HUI2 CAN

Price
$184999

Colour
Yellow

Transmission
Auto

Mileage
10

Fuel Type
Petrol

Engine Size
5800CC

Doors
2

Condition
Band New

Notes
Absolute ROCKET 🚀

Figure 1 Android App.
Images are for illustration purposes only. You may implement the design (colours, layout, images etc.) of the app as you wish especially the details activity. The ListView on the Main Activity must show the make, model, year and licence number of the vehicles.

Code and Visual Quality (10%)

At a very **minimum** the **main activity of** your app should **display** the **make, model, year and license number** of all vehicles in the database**.** The overall look and design is up to you. The images/icons used should be scaled appropriately, and the app should look reasonably well polished (**sensibly laid out, readable text, standard contrasting text, icons and background colours colours**). Your **code should be** sensibly **designed**, **laid out consistently** and well **commented throughout (i.e. android and server code)**.

Additional Functionality (up to 50%)

For the higher marks, more functionality will need to be added to the RESTful webservice on the server side and android application.

Server Additional Functionality (up to 25%)

- RESTful route to **INSERT** new Vehicles to the Database. **(5%)**
- RESTful route to **UPDATE** existing Vehicles to the Database. **(5%)**
- RESTful route to **DELETE** existing Vehicles to the Database. **(5%)**

  Add routes to your RESTful web service to support full CRUD database functionality. **For the full 5 marks all data must be exchanged in JSON format** (except DELETE). Meaning that when you are inserting or updating a new vehicle via the API, the client (android app) must send a JSON object containing all the vehicle information to the server. The server must then convert that JSON object to a Java Vehicle object and save/store/update it in the database.

- Implement an **API key** to protect your RESTful routes **(10%)**

  Implement an API key to limit access to your RESTful web service to only registered users with a valid API key. For the full marks this will require a users table to be added to the database and a mechanism for users to register themselves as developers and be issued a unique API key (randomly generated alpha numeric string). This API key should then be used in all app server communication. In instances where the API key is not sent as a parameter in a request made to the server the server should respond with an error and instruct the user to use their API key in order to access the RESTful services.

Android Application Additional Functionality (up to 25%)

- Add functionality to your app that allows the user to **INSERT** new vehicle records in to the server database using the REST API. *This can be achieved through an activity with EditTexts that the user fills in and then presses a button to submit that information to the server.* **(5%)**
- Add functionality to your app that allows the user to **UPDATE** existing vehicle records in to the server database using the REST API. *Functionally the update is VERY similar to the add vehicle, except the existing vehicle data will be displayed in the EditTexts when the update activity loads, then the user edits what they need to and submits the data to the server for updating.* **(5%)**
- Add functionality to your app that allows the user to **DELETE** existing vehicle records from the server database using the REST API. *One way this can be achieved is through a long press the ListView item that requires deleting which would then trigger a RESTful delete call to the server with the id of the Vehicle that needs to be deleted.* **(5%)**

- **AsyncTask**

  The task of talking to the server to send/retrieve data can 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 a big no no in the mobile app development world.

Improve matters by handling **ALL** network calls that send/retrieve data to/from your java server using the **AsyncTask** class, which will perform them in a **new thread in the background**. **(10%)**

** Remember to **remove the StrictMode** workaround discussed in lectures once threading is working. **

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** (OBS, 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 the functionality you wish to be marked on, demonstrate **all RESTful webservice functionality using a REST client** and all **android app functionality using the android emulator.** The total length of the screen cast be **no more than two to three 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.

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.

Java RESTful API Server 45%		**Android App 45%**		**Code and Visual Quality 10%**	
Server connects to the SQLite database and outputs all the vehicle information in JSON format with a **GET** request.	20 marks	App successfully connects to the server in order to retrieve and display the vehicle information on the MainActivity in an **ListView** (displaying vehicle make model, year **and** licence number). A vehicle in the ListView can be selected in order to view full details on a separate activity (**details activity**).	20 marks	All code submitted (server and android application) is well structured, laid out consistently and commented along with a screen cast demonstrating server and android app functionality.	10 marks
RESTful API Server implements a route which accepts http **POST** requests in order to add new vehicles to the database.	5 marks	App implements a functioning mechanism for **inserting** new vehicles to the server database.	5 marks		
RESTful API Server implements a route which accepts http **PUT** requests in order to update existing vehicle details in the database.	5 marks	App implements a functioning mechanism for **updating** existing vehicle details in the server database.	5 marks		
RESTful API Server implements a route which accepts http **DELETE** requests in order to delete existing vehicles from the database.	5 marks	App implements a functioning mechanism for **deleting** existing vehicles in the server database.	5 marks		
All the RESTful API server routes are protected with an API key. Server has a mechanism where a user can register as a developer in order to get an API key. As well as allow existing users/developers to login in order to retrieve their API key.	10 marks	**All** network operations are performed in an **AsyncTask**.	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.