Assignment 2 — Mobile client/server application

Due date:	11:00pm AEST, Friday Week 10	**ASSIGNMENT**
Weighting:	30%	
Length:	Less than 50 MB	**2**

Objectives

This assessment item relates to the learning outcome 1, 2, 3 and 4. More specifically to design and implement a complex mobile application.

The objective of this assignment is for students to:

Develop, test and maintain a mobile internet application using an integrated suite of mobile software development tools. More specifically on the client side *jQuery Mobile*, *JavaScript* and *HTML5* and *CSS* are used. On the server side we use *JavaScript* and the *node* server with various *node* packages. The *Mongo* database is used as a data store and that data is also stored locally on the mobile device. Using *JavaScript* store data in a *mongdb* database in a *MongoLab* cloud server using *node express* middleware.

Introduction

You are assigned the task of creating a data logger to capture experimental data in a mobile application that stores data in a local database. The app has fields to record data for each of five days of use of drone aircraft. When a day is selected, a page is shown to record data values for drones that are recorded by the app. A drone entry consists of a **date** and **log** data. When the **Save Log Entry** button is pressed these values are saved locally in the devices' **localStorage**. When the **Show Log Entries** button is pressed a related page is shown that lists all the date/time and drone entries. More details of these pages will be given in the sections below.

We will refer to our app as **DroneLogs**. The specification of this app is further refined here from that given in assignment 1. This app is to be tested using the **Safari**, **FireFox** or **Chrome** browser and tested on an Android or iPhone mobile device.

Client Side HTML / CSS / JavaScript Mobile Application

You are to implement this app using HTML5, CSS and JQueryMobile. The app is used by a company who has to collect data on who is using their drones over rolling 5 day periods. A number of individual drones exist and for each drone a unique ID number is used to identify it. For each individual drone the company needs to record data that can be used to document who flies the drone and from where the drone is flown.

Images for the page/views required to implement the assignment were given in assignment 1 and are further refined in the following figures. Please note that the illustrations are for reference only, and your actual pages will be based on the CSS style-guide in use on your actual mobile device.

Home page view

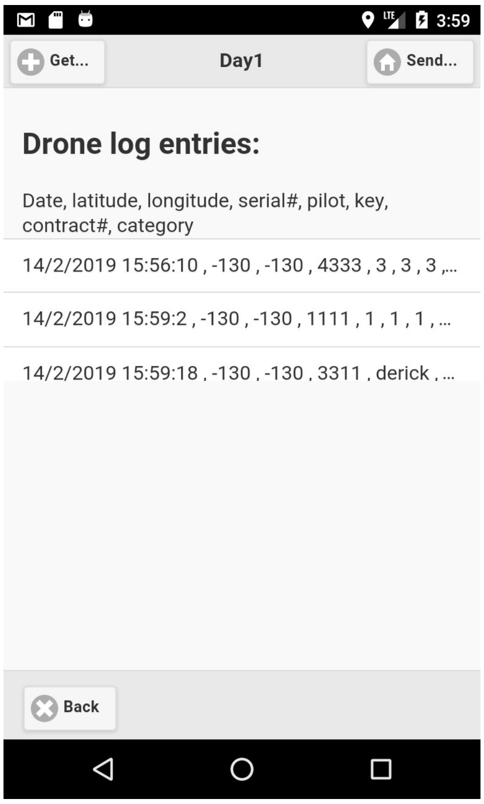
The home page view is the same as in assignment 1.

Drone view

The pages for entering the drone data are all the same as in assignment 1 and so are not discussed further.

Drone logs view

When the **Show logs** button in the drone's page header is pressed the current date/time and latitude/logitude are added to the data structure used to store the drone log values being recorded, as in assignment 1. The log entries are saved in the device's **localStorage**. The drones logs page is shown with all the locally saved logs listed, as shown in Fig. 1. Note the view now has a **Get …** button on the left of the header bar. Also note the button on the boom is now a Back button. Pressing this should take the user back to the previous page (Drone view).



Drone log entries:

Date, latitude, longitude, serial#, pilot, key, contract#, category

14/2/2019 15:56:10 , -130 , -130 , 4333 , 3 , 3 , 3 ,...

14/2/2019 15:59:2 , -130 , -130 , 1111 , 1 , 1 , 1 , ...

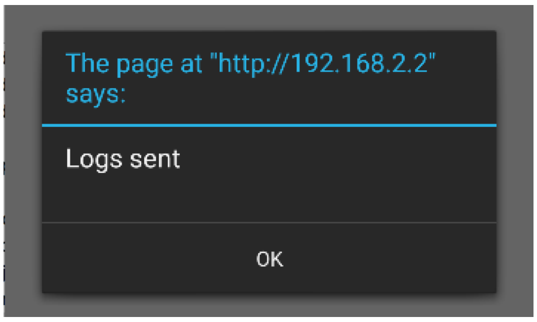
14/2/2019 15:59:18 , -130 , -130 , 3311 , derick , ...

Fig. 1: Drone day log page.

Send button

When the **Send…** button is tapped all the drone logs are sent to the local server (and saved in the file `logs/logs.dat`. The drone data is also saved to the cloud mongolab *mongdb* database set up for this purpose. You should provide a success or failure alert. In the success alert, show the data that has been sent. When a response is received another alert should indicate success or failure. When a drone's logs have been sent the drone local logs should be cleared from **localStorage** so that the drone page will not show the sent logs.

If the **Yes** button is pressed the **Send logs** dialog shown in Fig. 2 is presented and we return the drone view. If the **No** button is pressed we just return to the drone's view page.



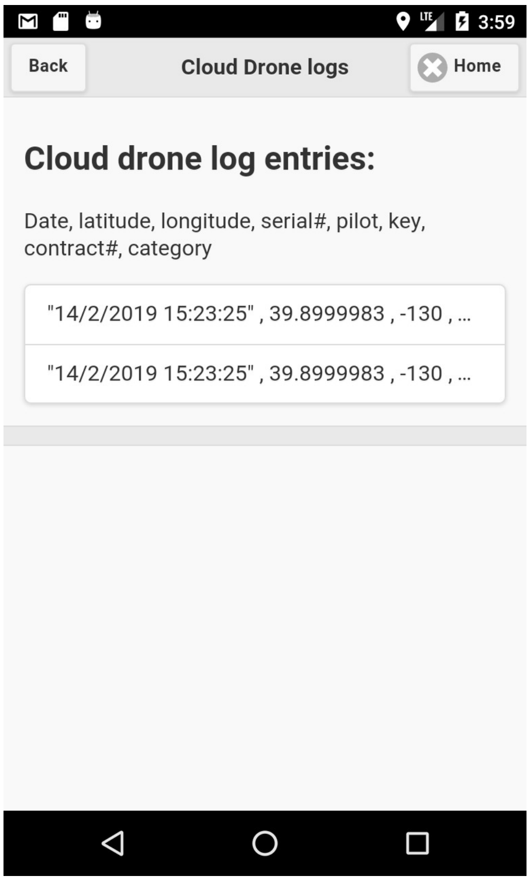
The page at "http://192.168.2.2" says:

Logs sent

Fig. 2: Logs sent confirmation.

Get logs button

The **Get logs** button is used to search the mongolab *mongdb* database for all database entries that match the day for the drone records (**Day 1** in the figure). The entries returned are shown on a new page as shown in Figure 3. The entries are to be shown below the **Cloud drone log entries:** label as shown in the figure 3. Again appropriate alerts need to be made when the request is sent and when a response is received. The **Back** button takes the user to the previous page and **Home** takes the user to the home page.


3:59

Back

Cloud Drone logs

Home

Cloud drone log entries:

Date, latitude, longitude, serial#, pilot, key, contract#, category

"14/2/2019 15:23:25" , 39.8999983 , -130 , ...

"14/2/2019 15:23:25" , 39.8999983 , -130 , ...

Fig. 3: Cloud Logs entries page.

Server Side: node+packages and JavaScript server script

Our user data scheme has the following fields:

- **day** – one of {Day1, Day2, Day3, Day4, Day5}
- **date** –date and time stamp of log
- **latitude** –latitude of drones starting location
- **longitude** – longitude of drones starting location
- **serial_no** – drones ID
- **pilot** drone pilot
- **key** – key used for encrypting communication with drone
- **contract_no**– contract number for drone
- **category** – user of drone

This data is to be stored in the mongolab *mongdb* database in a **drone_logs** collection. Entries are also to be echoed (written) to a file in the **./logs** directory of the local server in a file called **logs.dat**.

The server will have the following URL that provides requested services. The URL is based on **http://your.IP.address/drone/user/**.

Your web service API will support these actions:

search/:query– searches for users in the *mongoLabs* database and returns all logs with that **:query** value. **:query** will be one of the day values {**Day1, Day2, Day3, Day4** or **Day5**} to search for.

:drone/log – appends the drone entry to the local server file **./logs/logs.dat** and to the *mongoLabs* drone_log *mongdb* databases **logs** collection.

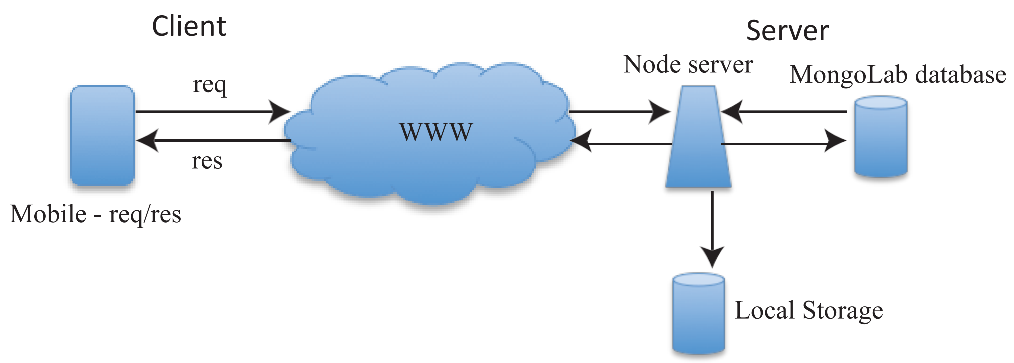
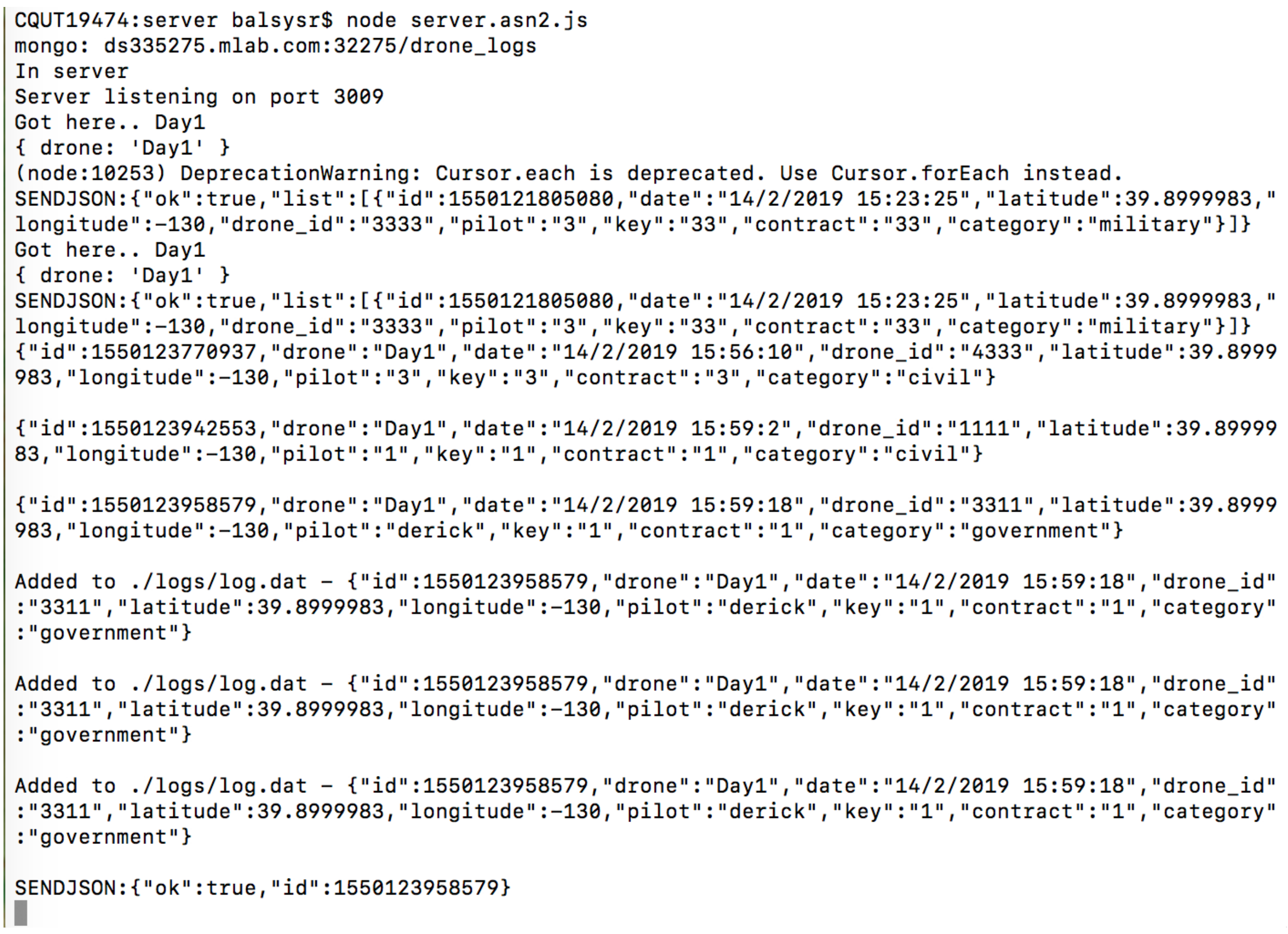


Fig 4. Message flow from App to local disk storage and Cloud database

Fig. 4 shows the message streams in the application. Ideally the mobile device POST's requests to the WWW and receives responses from the WWW. The *node* server listens for requests on a port. The requests data will be routed to the local *mongdb* database at *mongoLabs*. Requests for the information in the *mongdb* database would be returned to the *node* server for POST'ing back to the mobile device. The server should produce meaningful output each time a request is received or sent.

Your *node* server code will consist of a number of files; *server.js* will contain your business logic, *common.js* and *config.js* contains common utility functions and network configuration information. The *server.js* file will use **express** middleware to create a server and router to route the web service API to the handler code that writes the JASON data to your *mongdb mongoLabs* database, and returns data from this database to the mobile device. A sample of the kind of responses the server should produce is given in fig. 5



```
CQUT19474:server balsysr$ node server.asn2.js
mongo: ds335275.mlab.com:32275/drone_logs
In server
Server listening on port 3009
Got here.. Day1
{ drone: 'Day1' }
(node:10253) DeprecationWarning: Cursor.each is deprecated. Use Cursor.forEach instead.
SENDJSON:{"ok":true,"list":[{"id":1550121805080,"date":"14/2/2019 15:23:25","latitude":39.8999983,"
longitude":-130,"drone_id":"3333","pilot":"3","key":"33","contract":"33","category":"military"}]}
Got here.. Day1
{ drone: 'Day1' }
SENDJSON:{"ok":true,"list":[{"id":1550121805080,"date":"14/2/2019 15:23:25","latitude":39.8999983,"
longitude":-130,"drone_id":"3333","pilot":"3","key":"33","contract":"33","category":"military"}]}
{"id":1550123770937,"drone":"Day1","date":"14/2/2019 15:56:10","drone_id":"4333","latitude":39.8999
983,"longitude":-130,"pilot":"3","key":"3","contract":"3","category":"civil"}

{"id":1550123942553,"drone":"Day1","date":"14/2/2019 15:59:2","drone_id":"1111","latitude":39.89999
83,"longitude":-130,"pilot":"1","key":"1","contract":"1","category":"civil"}

{"id":1550123958579,"drone":"Day1","date":"14/2/2019 15:59:18","drone_id":"3311","latitude":39.8999
983,"longitude":-130,"pilot":"derick","key":"1","contract":"1","category":"government"}

Added to ./logs/log.dat - {"id":1550123958579,"drone":"Day1","date":"14/2/2019 15:59:18","drone_id"
:"3311","latitude":39.8999983,"longitude":-130,"pilot":"derick","key":"1","contract":"1","category"
:"government"}

Added to ./logs/log.dat - {"id":1550123958579,"drone":"Day1","date":"14/2/2019 15:59:18","drone_id"
:"3311","latitude":39.8999983,"longitude":-130,"pilot":"derick","key":"1","contract":"1","category"
:"government"}

Added to ./logs/log.dat - {"id":1550123958579,"drone":"Day1","date":"14/2/2019 15:59:18","drone_id"
:"3311","latitude":39.8999983,"longitude":-130,"pilot":"derick","key":"1","contract":"1","category"
:"government"}

SENDJSON:{"ok":true,"id":1550123958579}
```

Fig. 5 Sample server screen feedback

See the weekly lecture/workshops for weeks 8, 9 and 10 for more information on this assignment.

Required Documentation

You are also to prepare a Word document. Your word document should include an appropriate title page. Your word document should have sections that address the commentary, hardware and software requirements and the financial analysis given below

Plan for a Testing Strategy for your Mobile Web Site

Your document should:

- Discuss the selection of mobile browser models, mobile OS's and device manufacturers for your testing process
- Discuss whether testing is required on actual devices and what part emulators/simulators can play in this process
- Document functional and UI testing of the app.

Financial Case & Commentary of Application Features

Your document should:

- List the features you successfully implemented and those you were unable to successfully implement; you should describe the problem in a few sentences and also briefly describe anything you attempted to do to get it to work. Your approach to identifying and attempting to fix these bugs may gain you some partial credit for those features you were unable to implement.
- A description of any additional functionality you believe would be useful to add to this Website should be included. Explain what the features are and how they would help to improve the Website.
- Are there any ethical considerations in making this kind of service available?
- Discuss the economic/financial case for the proposed system.

  Assume this simple exercise leads you to consider developing a fully-fledged *DroneLogs* application (that is generalised to handle any kind of log data for any number of days) with increased functionality. Assume you model 3 cases; one with 10 users, one with 1,000 users and one with 10,000 users.

  - Assume staff development time is costed at $100/hour. Given how long it has taken you to develop this exercise what do you estimate technical development costs to be in hours? What are the development costs for the 3 models, assuming development time increase by 50% for each of the scenarios?
  - Database storage costs. If you use the *mongdb mongoLabs* service to store *droneLogs* entries, how much would this cost for your proposed user base, assuming 10 entries per month per user?
  - Discuss your marketing campaign for each of the three cases and estimate a cost for each of these.
  - When do you estimate you break even (costs=revenue) in the 3 scenarios? Comment on the results of this analysis.

Submission

You are required to submit your assignment electronically via the Moodle course website. The deliverable is a rar or zipped directory containing all the code and resources needed for testing. You must tar, rar or zip together:

- The directory containing all the files, folders, images required to test your *mobile* application
- Your Word document

The resulting gzip, rar or zip file should be submitted on the course website. Please note that you should use your student number as the name for your gzip, rar or zip file when uploading to Moodle so that all students work can be linked back to the author.

Assessment criteria – Ass2 - COIT20269 Mobile Web Apps

Student name :

Student Number :

Assignment Component	Criteria	Marks	Total
Cordova Phonegap Integration	The *DroneLogs* app is successfully implemented in Cordova Phonegap	**2**	
Client Side application			
Send/Get buttons	- Javascript and HTML files correct - Tap handler for Send sends data with feedback correctly - Tap handler for Get gets data with feedback correctly - Current logs page meets specification - Cloud drone logs page meets specifications - The data in the logs pages are formatted correctly	**6**	
Server Side application			
Config.js / config file	- The commonly used functions are all defined correctly - Cross site scripting is handled correctly	**2**	
DroneLogs.js	- Appropriate response messages are sent, and appropriate messages are written by the server as messages are routed (1 mark) - The drone_logs collection at MongoLabs is successfully connected, opened and updated (2 marks) - The log() methods appends each entry received to ./logs/logs.data on server (1 mark) - All entries for the given drone are returned in the response object when the search/:query is performed (2 marks) - The :drone/log request writes the data to the drone_logs user collection of the *mongdb* database (2 marks)	**8**	
Discussion of testing regime	- Discussion of selection of mobile models - Discussion of part emulators vs. the real devices play in testing - Document functional and UI testing of the app	**3**	
Commentary	- Successful/unsuccessful features - Additional functionality - Ethical considerations	**3**	
Financial case	- Time costing - Database costing - Marketing - Analysis - Synthesis	**5**	
General			
	- Feedback given as required - Use appropriate naming conventions - Adequate commenting - Correct grammar - Citation of references, copyright use	**1**	
Penalties			
	Total	**30**	

Lecturer Comments

Lecturer's Signature__________________________ **Date:**