Assignment 2 - Room Usage App
ENG1003, Semester 2, 2018

Due: Friday 12 October 2018 18:00
Worth: 24% of final mark

Aim

You will be producing a location–aware app for tracking usage statistics in rooms around Monash University. The purpose of this is to allow users to fill in a form for given rooms at different times of the day recording lighting, air-conditioning and room utilisation.

Background

You are part of a software development team in an organisation called *sustainAppility*, a group who specialises in apps that encourage careful use of resources and awareness of environmental impacts. Your organisation has recently secured the business of Monash University, who are looking to become greener and make better use of their resources. Your team has been tasked with developing a mobile web app for this purpose.

It is hoped that such an app could be distributed amongst their students to allow them to report on how rooms throughout their campus are being used, so that the organisation can make adjustments to heating, cooling and lighting. Monash University have given your team one month to prepare a prototype. If they are happy with this prototype they will be willing to hire your team to build and maintain a full system for the next five years.

Monash is the largest client that sustainAppility has attracted, so your boss is hoping to make a good impression. This contract could make or break the organisation… no pressure or anything.

What you are provided with

We have made available a skeleton version of the app. The skeleton provides you with several HTML pages and associated JavaScript and CSS files representing the structure of the app. You will need to implement the logic and functionality for each of these pages as described below. **You are NOT allowed to change the file names**.
The app skeleton contains 4 web pages that utilise MDL for their interface:

- *buildingStats.html*
- *index.html*
- *observations.html*
- *occupancy.html*

The app skeleton contains 1 CSS file in the css folder:

- *style.css* (included in all HTML files)

The app skeleton contains 5 JavaScript files in the js folder:

- *buildingStats.js* (included in *buildingStats.html*)
- *formView.js* (included in *index.html*)
- *observation.js* (included in *observation.html*)
- *shared.js* (included in all HTML pages)
- *utility.js* (included in all HTML pages, don't modify)
- *occupancy.js* (included in *occupancy.html*)

You will need to create some classes, which should be declared in the *shared.js* JavaScript file so that they can be used by all three HTML files.

In the app skeleton, these pages are mostly blank and include sample entries only to demonstrate how to navigate and communicate information from one page to another. Other than the *utility.js* file, you may modify these as you like, however you should not change the names of these files.

The app skeleton contains a *displayMessage* function (in *utility.js*) which you can use to display pop-up "toast" notifications at the bottom of the screen. You should not modify *utility.js* in your solution.

What you need to do

This assignment consists of multiple tasks, including programming, documentation, and use of software tools. It is strongly recommended that you practice pair programming rather than trying to split up the coding and attempting parts individually.

You will need to communicate effectively with your team while working on the assignment. Your individual use of Git, Asana, and Google Drive will affect your final marks on this assignment, and as with Assignment 1, your final marks will be scaled by your team contribution as determined by CATME feedback.

Finally, there will be a 15-minute team "client" presentation as well as an individual interview on your submitted code, both of which will occur during your week 12 practical class. This is described in a later section in this document.

Getting started

1. One team member should create an "assignment2" directory with the necessary initial directory structure in your team Git repository by committing the provided skeleton code (see the Skeleton code and also the "Submission" section below).
2. Set up an API key for `https://geocoder.opencagedata.com/`
3. Read and discuss the list of required functionality below and create Asana tasks for the necessary features.
4. Discuss and plan out how the functionality of the app will work, which can be done by drawing activity diagrams for the process flow of the app.
5. Draw the storyboard and wireframes for the entire app. (You will do this in your Week 8 prac activity.)
6. Draw the class diagram for the two main classes used in the app.
7. Based on your storyboard and wireframes, break down the work to be done and decide how you will tackle them.

Required Functionality

Feature 1: RoomUsage class

As part of this application you will need to implement a *RoomUsage* class to hold information about a specific room at a given time. The RoomUsage class should be defined in in shared.js so that this class is available on all pages. This class will need to include the following attributes:

Attribute name	**Description**	**Data type**	**Restrictions**
_roomNumber	Room number	string	Non-empty
_address	Building address	string	Non-empty
_lightsOn	Lights on	boolean	true; false
_heatingCoolingOn	Air Conditioning/ Heating on	boolean	true; false
_seatsUsed	Number of seats in use	number	Positive integer, <seatsTotal
_seatsTotal	Total number of seats in room	number	Positive integer
_timeChecked	Date/time usage checked	Date class instance	

Feature 2: RoomUsageList class

Create a *RoomUsageList* class to represent a list of room usage observations. This class should have a single attribute named *_roomList*, an array of *RoomUsage* class instances. It should have appropriate methods for adding and accessing observations, as well as producing the results required for Feature 7 (described below).

Feature 3: New Room Observation

The New Room Observation page (*index.html* and *formView.js*) has a form made up of a combination of MDL Text Fields, Buttons and Toggle elements, which allows users to enter data for a given room. This data corresponds to the information described by the **RoomUsage** class attributes above. We suggest that whenever you display building addresses in the app you display just the most relevant part of the address, i.e., up to the first comma, rather than the whole address which might include state and country.

When the “Clear” button is tapped, the app should clear/reset all fields in the form.

When the “Save” button is tapped, the app should validate the form to ensure that each field has the appropriate type of data and a sensible value. If it does, it should create an instance of the **RoomUsage** class with those values and add this to the *RoomUsageList* class instance and let the user know. Otherwise, it should show a useful error message to the user (the *div* with the ID *errorMessages* can be used for this).

Note: The time recorded in the room usage should be the time the class was constructed, i.e., the time when the Save button was tapped.

Feature 4: Location Tracking

While on the New Room Observation page, the app should use “geolocation” (see references) to determine the user’s current location. You should use this location information to make suggestions to the user about their address via a process called “reverse geocoding” (see references). When the “Automatically determine my address” checkbox is selected, the app should watch the user’s location and use reverse geocoding to automatically update the building address field.

You will be using the geocoder API below

`https://geocoder.opencagedata.com/`

To use this API you will need to set up an API key. When you make a JSONP request to this web service you will need to specify the callback function using the parameter `jsonp` e.g., `jsonp=yourFunctionName`. Also, when you specify the latitude and longitude, you will need to combine them together with a comma (not a plus) e.g., `q=139.3465,-42.2344` (the documentation is a little confusing).

Note: It is possible to receive results which are not 100% accurate for Monash campuses as the information in `geocoder.opencagedata.com/api` may not contain all the building addresses for Monash Clayton or Sunway.

Feature 5: Storing RoomUsageList data in localStorage

We want to ensure that room utilisation data is stored persistently in the web browser. When a new **RoomUsage** class instance is created, the app should add that to your instance of the **RoomUsageList** class and save these in local storage so that they can be loaded when pages of the app are loaded in future.

You should store data to and retrieve data using the key 'ENG1003-RoomUseList' in local storage.

Feature 6: Loading RoomUsage data from localStorage

Make sure that your code is able to recover the data from you stored to local storage (Feature 5) and reinstantiate it as a RoomUsageList class. This RoomUsageList class instance will need to be available to the code of all four of the pages of the app.

To save you from collecting a large number of observations in order to test the app, we have provided you with some JSON containing a number of saved observations. You can find this in the `skeleton/testData.json` file. You can copy the contents of this file and use the Chrome Web Inspector to set the appropriate *localStorage* entry to this JSON string. You should ensure your submission correctly loads this JSON.

Hint: As part of loading the *RoomUsageList* class you will need to load a number of instances of the *RoomUsage* class in the same way you did in the Week 7 workshop and prac activities.

Hint: When initialising the `_timeChecked` attribute of your RoomUsage class from the simple object version, you can pass the JSON string version of the date to the `Date` class constructor as an argument and it will correctly create the date.

Feature 7: Observations page

The Observations page (*observations.html* and *observations.js*) should display a list of all recorded observations in reverse order, i.e., with the most recent at the top.

For each observation the page should show the information contained in that *RoomUsage* instance. We provide some sample HTML that can be used as a template for each observation. (You can also display it differently if you choose.)

Feature 8: Deleting observations

The Observations page should provide a way to delete observations. Our sample HTML for observations in *observations.html* contains a delete button for each entry. When tapped, this button should cause the corresponding observation to be removed from the RoomUsageList and the displayed list of observations updated.

Hint: The easiest way to do this is when generating the HTML for each observation, generate a property of the delete button HTML that calls a function with the index of the observation within the array of all observations, e.g., `onclick="myDeleteFunc(47);"`. The index will be known to you at this point since you will be iterating through the array indices to display all the observations.

Feature 9: Searching the observations

The Observations page should allow you to enter a search term to filer the list of observations that are displayed. The skeleton HTML has a search icon on the top-right of the app's title bar. When tapped this icon expands to show a search text field.

When the user makes a change to this text field (ID: *searchField*) you should call your function that generates the observations list HTML (that you already call when the page loads), but only have it output observations where the search term is part of the building address or room. Case should be ignored.

Hint: The *oninput* property of text field DOM elements can be used to set a callback function that is triggered whenever the contents of the field is changed, i.e., on each key press.

Hint: You can ignore case of search terms by converting the search term to lower case (like in our Palindrome workshop example) and looking for this within a lower case version of the building address. `String.indexOf` is useful for this later task.

Feature 10: Bucketing room observations

The next two features will require us to have a way of splitting up the full observations list into a number of *buckets*. You should add a method, *aggregateBy* to the *RoomUsageList* class.

The *aggregateBy* method should have a single parameter, which will itself be a function. This function returns an aggregation key given a *RoomUsage* class instance, and is used to specify how the bucketing is performed. For example, if you passed a function that returned the room string for a given *RoomUsage* instance, then the observations would be split into separate lists, one list for each different room.

The *aggregateBy* method should return an *object* that has a property for each bucket where the property names are the key values that were sorted by (e.g., room number string) and the property values are *RoomUsageList* instances for observations that correspond to that key value.

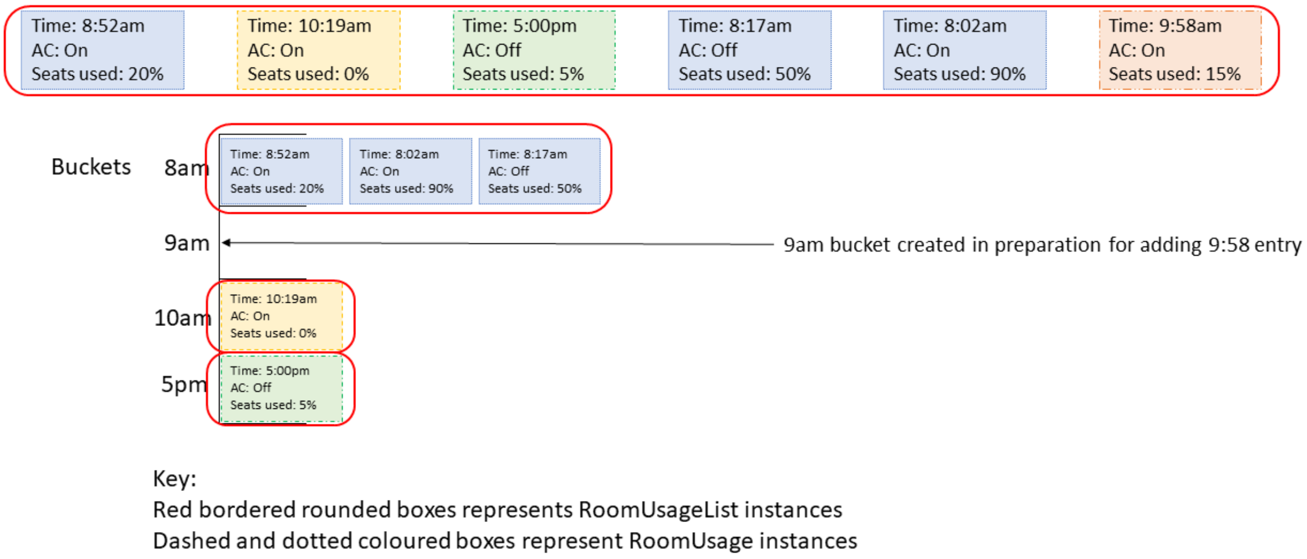

Figure 1: RoomUseList instances in buckets for each hour in the data set

Hint: The video on using JavaScript objects as dictionaries (from Assignment 1) will be useful here. You may also need the `Object.hasOwnProperty` method so you can choose between adding a new bucket to your return object versus adding an additional observation to that bucket.

Feature 11: Worst occupancy by hour

The Occupancy page (*occupancy.html* and *occupancy.js*) should bucket the observations by hour of the day, and for any hours of the day between (and including) 8am and 6pm should display the worst five room observations in terms of occupancy (how many seat are used in the room) or all observations if there are less than five in that bucket. It need not display entries for hours in this range that have no room observations.

The *observations.html* file includes sample HTML that could be used to show this information. It is fine to use this, or you may create your own if you wish.

Hint: You will need to use the *aggregateBy* method you created for the previous feature, and the `Date.getHours` method will be useful to give you key needed for this bucketing.

Hint: To order observations within the buckets, you should investigate the `Array.sort` method which can be passed a comparator function that takes as arguments two adjacent array elements (*RoomUsage* class instances) and returns whether the first is equal to, less than or greater than the second in terms of the desired order.

Feature 12: Building averages

The Building Stats page (*buildingStats.html* and *buildingStats.js*) should bucket the observations by building, and for all buildings with observations, display the building name and statistics on all observations for that building. These statistics should include:

- The number of observations
- The number of wasteful observations (those where there are no seats used but the lights or heating/cooling are on)
- The average seat utilisation (total seats used across all observations divided by the total number of seats across all observations)
- The average lights utilisation (the number of observations with lights on divided by the number of observations)
- The average heating/cooling utilisation (the number of observations with heating/cooling on divided by the number of observations)

The *buildingStats.html* file includes sample HTML that could be used to show this information. It is fine to use this, or you may create your own if you wish.

Note: The order of buildings in the output is not important.

Hint: You will again need to use the *aggregateBy* method you created for the previous feature.

Feature 13: Buildings with wasteful observations

Finally, the Building Stats page should visually highlight somehow the entries for buildings that have at least one wasteful observation.

The Programming component of this assignment is worth 12% of your unit mark.

Your team can modify or extend your app as you like, provided it has the required functionality.

Technical Documentation

Your team should produce three pieces of technical documentation for your app. The documents should be submitted as A4 size PDF files (font size 12).

- A basic Project Management Plan. This is internal documentation detailing:
  - Project aim and scope
  - Team members, and their responsibilities
  - How you are going to do the project, e.g., team meetings, minutes, handling communication of changes to the code, processes
  - How you are going to handle any risks which could occur over the life of the project
  - A brief overview of the design of the app.
  - A class diagram detailing:
    - All classes used in the code including all their attributes, methods, and class name (excluding API classes),
    - The relationship(s) between different classes and multiplicities for these relationships

      This should be an A4 PDF page in landscape orientation

      You should prepare this diagram on the computer (rather than by hand) e.g. Visio, Powerpoint, Google Drawings, Lucidchart, etc.
  - Any other information you would want to give to a new team member

    See Project Management Plan document template on Moodle for what should be included.
- A short User Guide. This is external documentation for users that detail:
  - How to get started using the app, with screenshots
  - Any known bugs or limitations

    See User Guide document template on Moodle for what should be included.

Your team will be assessed based on the quality of these documents. This will be worth 6% of your mark for the unit.

Presentation

This assignment includes a presentation component worth 6% of your unit mark.

Your team is now very close to the deadline for the project and a representative for Monash University, Susy Tayn How, has contacted your team to organise for you to present and demonstrate the app in front of her and the entire Building Management team at Monash University. While it is not explicit, it has been implied that this will decide whether your team is contracted to perform future work for Monash University.

In your week 12 prac class your team will deliver a formal client presentation. It will be based on the app you produced for Assignment 2. You should ensure that your presentation isnt́ framed as a university assignment, but a formal project. Note that the purpose of this is not to ‘sell’ the app, but rather to convince the client that you met the requirements. As with any good presentation, it should be prepared well in advance of the due date (including any visual aids) and it should be well rehearsed as a group and individually.

Format

Each student team will deliver a 15-minute oral presentation (in prac class) describing and demonstrating the functionality of their app and detailing any limitations of the application. Every member of the team should present for 3-4 minutes.

- The target audience for this presentation is the client.
- This presentation would be delivered in a formal business setting and so all team members should be dressed appropriately.
- This presentation must discuss the structure and functionality of the application as well as any design decisions made.

Testing the app

Upload your code to the ENG1003 server using the assignment upload page. The uploader can be found here:
`https://eng1003.monash/uploader/`
Once you have uploaded the code, you can view the uploaded assignment by selecting the appropriate assignment from the dropdown list on the assignment viewer. The viewer can be found here:
`https://eng1003.monash/view`
You can easily test features 7 and 8 using the testData.json file included in the skeleton ***without needing to perform data entry*** for lots of different rooms. You can also generate your own test cases which follow this format.

Resources

- `https://geocoder.opencagedata.com/api`
- `https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API`

Submission

Your team should submit their final version of the application online via Moodle. You must submit the following:

- A zip file of your Assignment 2 folder named based on your team (e.g., "Team014.zip").
  This should contain your code and documents with the folder structure below.
  This should be a ZIP file and not any other kind of compressed folder (e.g. .rar, .7zip, .tar).

Please ensure that your Assignment 2 folder (and Git repository) use the following structure:

```
assignment2
 ├─ code
 │   ├─ css .4 style.css .3 js
 │   │   ├─ buildingStats.js
 │   │   ├─ formView.js
 │   │   ├─ observations.js
 │   │   ├─ occupancy.js
 │   │   ├─ shared.js
 │   │   └─ utility.js
 │   ├─ buildingStats.html
 │   ├─ index.html
 │   ├─ observations.html
 │   └─ occupancy.html
 └─ docs
     ├─ ProjectManagementPlan.pdf
     └─ UserGuide.pdf
```

The submission should be uploaded by the team leader and must be finalised by Friday 12 October 2018 18:00 (local time). *Please note: Your entire team needs to accept the assignment submission statement individually on Moodle.*

You also need to individually complete the CATME peer assessment survey as described below.

Your assignment will be assessed based on the contents of the Assignment 2 folder you submit via Moodle. You should ensure that the copy of the assignment submitted to Moodle is the final version that exists in your Git repository. We will use the same phones as your team phone when marking your assignments.
Your presentation will be given during your practical classes in week 12 (15th–19th October 2018).

Marking criteria

This assignment consists of several component assessment items with the following associated marks (percentages of total marks for the unit):

- App code and functionality — 12% (group)
- Production of technical documentation — 6% (group)
- Presentation - individual component — 3% (individual)
- Presentation - Team component — 3% (group)
- Use of Git (Code), Asana (Task Management) and Google Drive (Documentation) — (moderating factor to assignment)
- Interview — (moderating factor to code)
- Peer Assessment — (moderating factor to assignment)

App code and technical documentation

Assessment criteria:

- Required functionality and correct behaviour of the produced app
- Quality of app source code, including overall structure and code documentation
- Clarity and quality of individual oral presentations
- Structure, appropriateness, and level of team-client presentation
- Participation and appropriate use of tools for team software development, including communication style
- Appropriateness of technical documentation, including structure, completeness and communication quality

You will be marked as a group, however your individual marks will be subject to peer review moderation based on CATME feedback and scaling factors.

A detailed marking rubric will be available on the unit Moodle page.

CATME Peer assessment

You are expected to work together as a team on this assignment and contribute roughly equal amounts of work. Peer assessment will be conducted via the CATME online system. You will receive email reminders at the appropriate time.

Not completing the CATME peer assessment component may result in a score of zero for the assignment.

Do:

- Give your teammates accurate and honest feedback for improvement
- Leave a short comment at the end of the survey to justify your rating
- If there are issues/problems, raise them with your team early
- Contact your demonstrators if the problems cannot be solved amongst yourselves

Do NOT:

- Opt out of this process or give each person the same rating
- Make an agreement amongst your team to give the same range of mark

Contribution through use of collaborative tools

Each team member will be individually assessed on their use of three tools for collaborative software development:

- Git (and GitKraken desktop client) for managing revisions of the app source, and handling commits by multiple team members.
- Asana for team communication, project management and issue tracking.
- Google Drive for document authoring.

The history of your contribution over the **entire period of the assignment**, on Git, Asana and Google Drive will be individually considered. For the use of each of these tools you will be given a score depending on your observed level of contribution. Students with less than the acceptable level of contribution for will incur a penalty to their assignment mark:

Score	**Penalty**
No contribution	10%
Some contribution	5%
Appropriate contribution	0%

Note: It is not enough to just use these tools for some dummy actions just prior to submission. This will not be counted. It is expected that you will use these tools regularly throughout the term of the assignment. You must give your demonstrator access to your team Asana workspace and Google Drive folder for Assignment 2.

Assignment code interview

During your week 12 prac class your demonstrator will spend a few minutes interviewing each team member to individually gauge the student's personal understanding of your Assignment 2 code. The purpose of this is to ensure that each member of a team has contributed to the assignment and understands the code submitted by the team in their name.

You will be assigned a score based on your interview, and your code mark will be penalised if you are unable to explain your team's submission:

Category	**Description**	**Penalty**
No understanding	The student has not prepared, cannot answer even the most basic questions and likely has not even seen the code before.	100%
Trivial understanding	The student may have seen the code before and can answer something partially relevant or correct to a question but they clearly can't engage in a serious discussion of the code.	30%
Selective understanding	The student gives answers that are partially correct or can answer questions about one area correctly but another not at all. The student has not prepared sufficiently.	20%
Good understanding	The student is reasonably well prepared and can consistently provide answers that are mostly correct, possibly with some prompting. The student may lack confidence or speed in answering.	10%
Complete understanding	The student has clearly prepared and understands the code. They can answer questions correctly and concisely with little to no prompting.	0%

Presentation

Students are marked individually for this assignment on their presentation skills Assessment criteria:

- Voice is of appropriate volume, speed and enthusiasm
- Language is appropriate for a formal context and jargon is only used where necessary (and explained if used)
- Eye contact is consistent and covers most of the audience
- Body language complements the presentation
- Explanations are clear and visual aids used appropriately

Teams are assessed on the structure and management of the presentation using the following criteria:

- Introduction clearly identifies the members of the team, what is to be spoken about, and who is talking about what
- Conclusion clearly closes the presentation and refers to previously covered content
- Material included is relevant, justified and of high quality
- Speakers are given an equal share of time and are smoothly transitioned between
- The presentation adheres to the required time limit and the time spent on a section is appropriate for the complexity of that section
- Attire is appropriate for a formal setting
- Appropriate use of visual aids

Final marks will be a combination of **both** the individual and team marks.

Other information

Where to get help

There will be a FAQ posted in Moodle and updated periodically. You can also ask questions about the assignment on the General Discussion Forum on the unit's Moodle page. This is the preferred venue for assignment clarification-type questions. You should check this forum (and the News forum) regularly, as the responses of the teaching staff are "official" and can constitute amendments or additions to the assignment specification. Before asking for a clarification, please look at the FAQ and forum.

Plagiarism and collusion

Cheating, Plagiarism and collusion are serious academic offenses at Monash University. Students must not share their team's work with any student outside of their team. Students should consult the policies linked below for more information.

`https://www.monash.edu/students/academic/policies/academic-integrity`
`https://eng.monash.edu.au/current-students/cheating-and-plagiarism.html`

See also the video linked on the Moodle page under the Assignment block.

Students involved in collusion or plagiarism will be subject to disciplinary penalties, which can include:

- The work not being assessed
- A zero grade for the unit
- Suspension from the University
- Exclusion from the University

You are required to reference code that has been obtained or provided by other sources (i.e. online), including formulas for calculating. This should be done within a comment above the code.

Late submissions/Extensions

We do not grant extensions on the assignment unless the majority of a team has been affected substantially over the period of the assignment by factors out of your control, as outlined in the special consideration form. Such special consideration applications should be made to the unit email address with a completed form and supporting documentation within two business days of the assignment deadline for each affected member.
`http://www.monash.edu/exams/changes/special-consideration`
Without an approved extension, late submissions are not accepted.

Unavailable team members

If team members are missing on the day of the presentation, the remaining members should proceed without them. Missing team members will receive a mark of zero unless they are granted special consideration. Such special consideration applications should be made to the unit email address with a completed form and supporting documentation within two business days of the presentation date.

`http://www.monash.edu/exams/changes/special-consideration`

You must also inform your team members ahead of time if you will be absent on the day of the presentation. If your team has less than three members presenting, notify the unit staff.