Assignment 2 —
Weather tracking app

MCD4290, Trimester 1, 2017
Due: Tuesday May 16th, 11:55PM (local time)
Worth: 25% of final mark

Aim

Many people use their mobile phones to look up weather conditions in their current location or other places of interest. Have you ever wondered how exactly these apps work? In this assignment you will build a location-aware app that utilises an online forecasting service to present to the user current and historic weather information for locations of interest.

Background

The State Emergency Service (SES) is a volunteer organisation providing assistance during emergency situations (such as floods, storms, building damage, etc.). In order to better protect and empower the Australian public the SES has put out a request for tender to hundreds of organisations to produce a weather app. The goal of this app is to keep the public informed of current weather and in particular any weather alerts relevant. Your team is representing a company called *'WetHair'* in the tender process with the goal of securing a contract.

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.

The app skeleton contains three web pages that utilise MDL for their interface:

- Location List page (main index page)
- Add Location page
- View Location page

In the app skeleton, these pages are mostly blank and include sample entries only to demonstrate how to change from one page to another. You may modify these as you like, however you should not change the names of these files.

What you need to do

This assignment consists of multiple tasks, including programming, documentation, and use of software tools. For the programming tasks, we suggest you complete each step together as a team before moving onto the next one.

We strongly suggest that you work on the programming tasks together as a team, or at least in pairs, rather than trying to split up the coding and attempting parts individually.

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. Someone in your team needs to sign up for an account on darksky.net, the web service you will use for weather data (see "Resources" section below). You will need an API key to call the darksky.net JavaScript API.
3. Your team should read and discuss the list of required functionality below and create Asana tasks for the necessary features. You may need to think about and break these down into further subtasks.
4. Consider usability principles when designing the application interface and behaviour.

Required functionality

Launch page ("Locations List")

The app should launch to page that displays a list of saved locations that have been added by the user. This list will initially be empty. These user-specified locations should persist between app launches, that is, be stored in the browser's local storage. A "location" consists of a nickname, a latitude and a longitude.

Each entry in the list should display the location nickname. For each location the current day's low and high temperature and condition summary (as text or icon) should also be shown. When first loading the app this information might not be available. Hence, the page should show that the weather for each entry is loading until the required data becomes available. You should use the `LocationWeatherCache` class to request weather information (see below).

This page should display a header bar at the top with the title "Locations". This header bar should have a "+" button on the far right, which when tapped opens the Add Location page.

Tapping on any entry in the Locations List should open the Location Weather page for that location.

Add Location page

This page should have a header bar at the top with the title “Add location”. It should show two text fields. The first text field should allow the user to type an address (street address or just city or town). The second text field should allow them to enter a nickname for the location, e.g., “Home” or “Work”. Aside from the text fields, this page should display a map and button titled “Add location”.

The expected workflow for this page is that the user taps the location field interface and type in an address. The app should utilise the Google Geocoding API (see “Resources” section below) to look up the GPS coordinates for the entered address and display the location on the map. The app should display the formatted address returned by the geocoding API as an annotation on the map. The map and address annotation is intended to allow the user to check the address was recognised as the desired location. Next, the user can optionally enter a nickname for the location. Then, if the user taps the “Add location” button the geocoded GPS location should be added to the `LocationWeatherCache` object (and consequently saved to localStorage) and the user returned to the Locations List page. If no nickname is specified by the user, the formatted address from the geocoding request should be used as the name of the location.

Note: If there is no valid GPS position for a location entered by the user, then nothing should be added to the Locations List and an error can be shown to the user.

Location Weather page

The Location Weather page should show weather information for the selected location. The top of this page should show a header bar with the nickname for the active location. The page should also include the following:

- A map displaying the active location,
- The date that the displayed weather applies to,
- A date selection slider,
- A summary of the weather for the particular date, and
- A “Remove this location” button

The date should be displayed in the “YYYY-MM-DD” format. We have provided a new method for the Date class called `simpleDateString` which returns a String formatted in this way.

The slider should be able to be dragged to change the displayed date, and the displayed weather. The slider should have 30 positions, representing the current day and the previous 29 days. The slider position should initially be set to the far right, i.e., the current day. The change to the selected date should happen in response to the user dragging the slider control, rather than waiting until the user has finished dragging.

The map should be centred on the selected location and should highlight the location in some way.

The weather information displayed for the selected date and location should include:

- A human-readable text summary of current conditions, e.g., “Partly Cloudy”
- Minimum temperature in ℃
- Maximum temperature in ℃
- Current humidity in %
- Current wind speed in km/h
- *Anything else you want to show.*

Like the Locations List, this page might need to display weather information which is not yet available. In this case it should show that the weather information is loading and this information should be automatically filled in once the info is available. You should use the `LocationWeatherCache` class to request weather information from the darksky.net API (see below).

If the user taps the “Remove this location” button then location being viewed should be removed from the `LocationWeatherCache` (and the change saved to localStorage) and the user should be returned to the Locations List page.

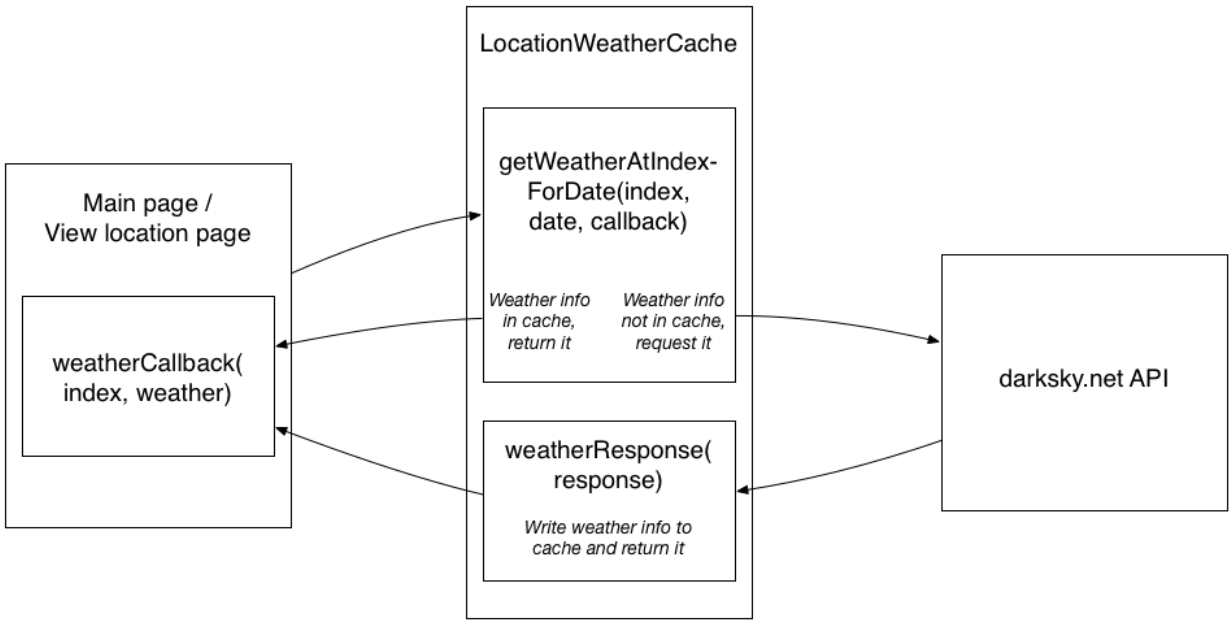
LocationWeatherCache class

This `locationWeatherCache.js` file in the skeleton contains a `LocationWeatherCache` class. This class includes method names but the methods themselves still need to be filled in. This file should not be dependant on any of the rest of the code of the app. This file will be included in the three web pages of the app, making the class available throughout the app.

The purpose of the `LocationWeatherCache` class is to provide a simple way to access weather information (current day or historic) for particular locations. This class should act as a cache for the darksky.net API. That is, when the app requests the weather for a particular location and date, the app should check the cache and return this information via a callback if it is stored locally by the class, otherwise it should request this data from the API, and then store the result in in the class and notify the app via a callback when it receives the information.

This class should have a private attribute called locations, an array that stores a list of location objects. This array should initially be empty. Each location object should have four properties: `nickname`, `latitude`, `longitude` and `forecasts`. The `forecasts` property should contain an object. This `forecasts` object should contain a property for each cached weather forecast. The property name should be a combination of the location and date, in the form of “{lat},{lng},{date}”, e.g., “46.0617697,12.078846800000065,2016-04-01T12:00:00” (the same format used for the darksky.net request). The property value should be the daily weather forecast object returned by darksky.net. (Hint: the `hasOwnProperty` method can be used to check if the forecasts object has weather for a particular location and date.)

The instance of the `LocationWeatherCache` class should be persistent, i.e, the information it stores should be preserved between app launches. This means storing it to Local Storage when the data changes. You will need to implement an `initialiseFromPDO` method for the `LocalWeatherCache` class which updates the class instance from a PDO object. You would get this PDO object by parsing JSON retrieved from Local Storage when loading the app.



The `getWeatherAtIndexForDate` method is the way for other code (Locations List page or Location Weather page) to request weather information for a location (specified by "index") on a particular date. The third parameter `callback` is the function that should be called with the weather (when it is available). This can't always be returned immediately because it may need to be fetched from darksky.net and could take an unknown amount of time.

There will be two cases for this method:

1. The weather information is already available in the cache (i.e., it is already in the `forecasts` attribute for the specified location. If so, the `callback` function should be called immediately with the index and weather information.
2. The weather information is not available in the cache. This method should request the weather via JSONP specifying the `locationWeatherCache.weatherResponse` method as a callback for this request (so the response can be stored in the cache). The `locationWeatherCache.weatherResponse` method will be called when the darksky.net request returns. This method should first save the weather information (the 'daily' weather data point) to the cache (`forecasts` property). It should then invoke the callback function originally specified as the third argument to `getWeatherAtIndexForDate` method. Obviously `callback` will not be available in this method, hence it will need to have been stored in the `callbacks` attribute.

Some more notes on the `getWeatherAtIndexForDate` method follow.

- This method takes three arguments: the index of the location, a JavaScript Date instance representing the requested date, and a callback function to be called when the weather is returned (see comments in the `locationWeatherCache.js` file).
- The time component of the date can be ignored. The app cares only about dates at daily granularity. We provide you a `Date.darkSkyDateString` method which works on a Date instance and returns a string formatted exactly for use with the darksky.net API.
- By default, the darksky.net API returns a lot of data we don't need. It is just the data point in the daily property that we are interested in. Hence the `minutely`, `hourly` and `currently` data blocks should be ignored, which can be done by including the exclude field in the API call query string (see the darksky.net API documentation).
- The user may specify different callback functions to be invoked for different location weather requests. In order to cope with this possibility it is necessary for you to temporarily store these callback function references in the `callbacks` private attribute of the class. These are only useful until the query has completed, and don't need to be serialised to local storage.

The `locationWeatherCache.js` file should contain a `loadLocations` function. When invoked, this function should create a single `LocationWeatherCache` class instance in a global variable named `locationWeatherCache`. It should then check local storage for an existing cache object. If there is one it should initialise the `locationWeatherCache` object from the serialised version in local storage. This function should be called at the end of the `locationWeatherCache.js` file. Since this file is included in all pages of the app, this ensures that the cache will be loaded from local storage and always be available.

The `locationWeatherCache.js` file should contain a `saveLocations` function. When invoked, this function should serialise the `locationWeatherCache` class instance to local storage.

General notes

Beyond providing the above functionality you should feel free further customise the layout, style and behaviour of the app.

The programming tasks together are worth 10% of your unit mark.

Technical documentation

Your team should produce two short pieces of technical documentation for your app:

- 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
    - Any other information you would want to give to a new team member
- User Guide. This is external documentation detailing:
    - How to use the various features of the app, with screenshots
    - Any known bugs or limitations
- You may include other documentation if you feel it is relevant

The technical documentation tasks are together are worth 9% of your unit mark.

Presentation

You should ensure that your presentation explains the features of the app. It should go into some high-level detail of how the app works and describe any specialised hardware required. You should talk enough about architecture to explain how the app might be extended or what additional features could conceivably be built on top of it.

Note: Your job is not to “sell” the app to the audience, but rather you are there as software engineers to explain the current state and capabilities of your app.

This presentation will be a formal shared talk delivered in front of your prac class. It will be based on the app you produced for Assignment 2. As with any good presentation, it should be prepared well in advance of the due date (including any visual aides) and it should be well rehearsed as a group.

Format

Each student team will deliver a 15 minute oral presentation (in prac class) describing and demonstrating their app and detailing any issues. Every member of the team should present for 3-5 minutes.

- The target audience for this presentation is a potential new client for the project.
- This presentation will be delivered in a business setting and so all team members should be dressed appropriately
- This presentation must discuss the design, structure and functionality of the application

The presentation component is worth 6% of your unit 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 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 the Assignment 2 code. The purpose of this activity is to verify academic integrity, i.e., 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 assignment mark will be penalised if your are unable to explain your team’s submission:

Score	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 provide answers that are mostly correct consistently. With prompting the student can provide partially correct answers. 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%

Testing the app

The main part of the app can be tested from a local directory with Desktop Chrome. Testing the GPS functionality requires the app to be uploaded to a web server and run on the team phone.

In order to test your app you should upload the source files for you app into your team directory on the ENG1003 server using the Assignment Uploaders or an SFTP client (see the Assignment 2 block on the unit Moodle page for details).

Resources

- http://www.getmdl.io/
  (MDL: Material Design Lite documentation)
- https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/Using_geolocation
  (Mozilla Developer Network: Reading the GPS sensor)
- https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Date
  (Mozilla Developer Network: Date documentation)
- https://darksky.net/dev/docs
  (The Dark Sky Forecast AI documentation)
- https://developers.google.com/maps/documentation/javascript/tutorial
  (Google Maps JavaScript API)
- https://developers.google.com/maps/documentation/javascript/geocoding
  (Google Maps JavaScript API: Geocoding Service)

Submission

You must your submit your assignment with the folder structure and file naming scheme shown below:

```
assignment2/
  code/
    js/
      mainPage.js                (Main page - Locations List)
      addLocationPage.js         (Add Location page)
      viewLocationPage.js        (View Location page)
      locationWeatherCache.js   (LocationWeatherCache class)
    css/
      weatherapp.css
    images/
      [...]
    addLocation.html          (Add Location page)
    viewLocation.html         (View Location page)
    index.html                (Main page - Locations List)
  docs/
    UserGuide.pdf
    ProjectManagementPlan.pdf
    GroupAssignmentCoversheet.pdf
```

These should be zipped up with the team name as the filename, e.g., “Team014.zip”. The contents of the zip file submitted to Moodle must match the content on the “Assignment2” folder in your team’s Git repository. The submission should be uploaded by the team leader by the assignment due date.

Your presentation will be during your practical classes in Week 12 (15th–19th May).

Each team member must individually complete the

- CATME peer assessment survey

as described in the “Peer Assessment” section below.

Your app will be assessed based on the version of the code you submit. We will upload it to the server and run it on the same type of phone as your team smartphone.

Late submissions

We do not accept late submissions without special consideration. Such special consideration applications should be made by submitted a complete special consideration form and supporting documentation within two business days of the assignment deadline to the Monash College reception at 49 Rainforest Walk.

Policy:
https://www.monashcollege.edu.au/__data/assets/pdf_file/0006/17097/dip-special-consideration-policy.pdf

Procedure:
https://www.monashcollege.edu.au/__data/assets/pdf_file/0006/567546/dip-special-consideration-procedure.pdf

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 the same way as late submissions (as above).

You must also inform your teammates if you will be absent on the day of the presentation.

Plagiarism

Cheating and plagiarism are serious academic offenses at Monash College. 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.monashcollege.edu.au/__data/assets/pdf_file/0010/17101/dip-assessment-policy.pdf
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 College and University
- Exclusion from the College and University

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

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 — 10% (group)
- Production of technical documentation — 9% (group)
- Presentation - individual component — 3% (individual)
- Presentation - Team component — 3% (group)
- Use of appropriate tools — (used to calculate individual contribution factor)

Assessment criteria:

- Required functionality and correct behaviour of the produced app
- Quality of app source code, including code documentation and overall structure
- 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.

Where to get help

You can 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 forum.