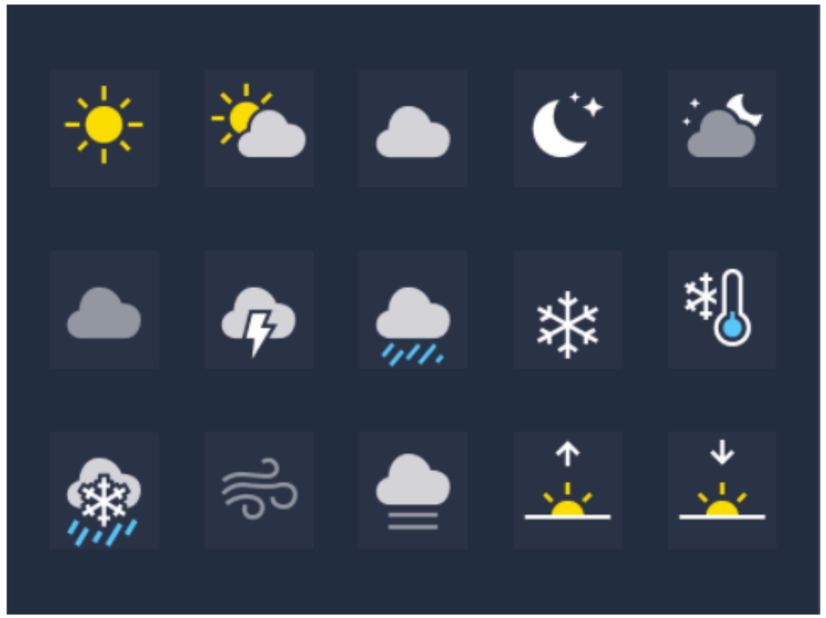
Project 2 Specification



Specification for Assignment 2

Agenda

Assignment Concept

Grading Bands

Grading Rubric

Where to start?

Schedule

Assignment Concept

- Owners of a consumer level weather station purchase and install a weather station kit.

- Periodically members submit weather reports from their station to a web site, capturing readings from the station at a specific time

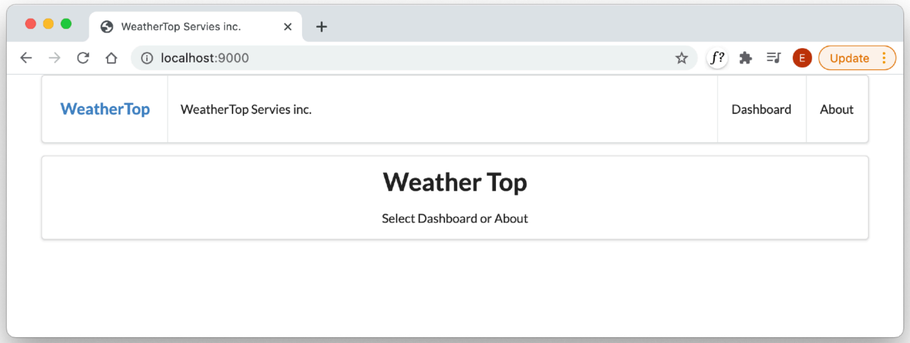
- The application displays weather analytics for the station

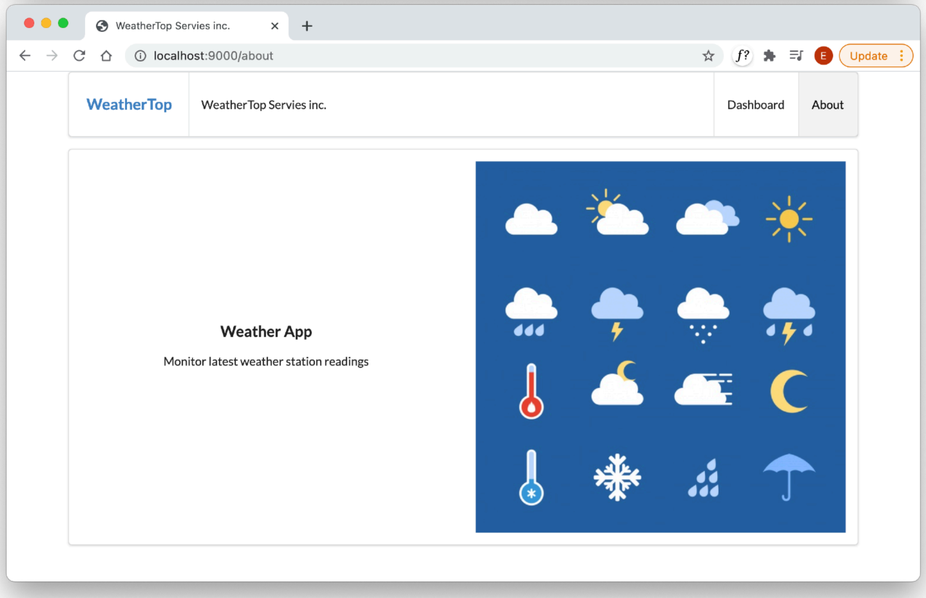
- The owner may own multiple stations

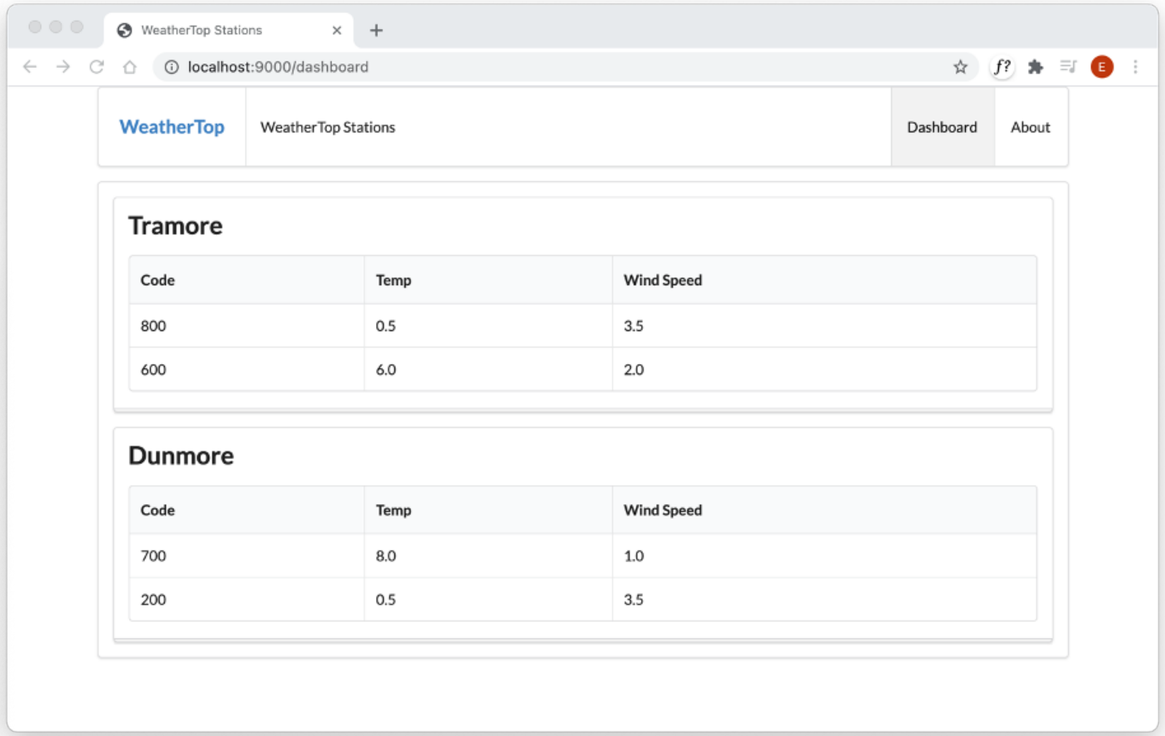
Grading Bands

Version	Package Label	Grade Range
1	**Starter**	0%-20%
2	**Baseline**	20%-45%
3	**Release 1**	45%-60%
4	**Release 2**	60%-70%
5	**Release 3**	70%-100%

	Reading	Station	Member	Features	Code
Starter	Code Temp Wind Speed	Station Name	None	Load and display stations + their readings from Yaml file	Zipped archive

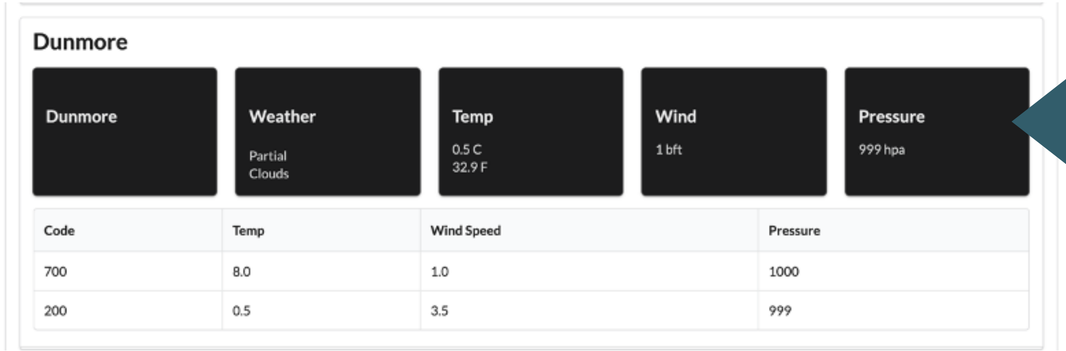







```
Reading(r1):
  code: 800
  temperature: 0.5
  windSpeed: 3.5

Reading(r2):
  code: 600
  temperature: 6.0
  windSpeed: 2

Reading(r3):
  code: 700
  temperature: 8.0
  windSpeed: 1

Reading(r4):
  code: 200
  temperature: 0.5
  windSpeed: 3.5

Station(s1):
  name: Tramore
  readings:
    - r1
    - r2

Station(s2):
  name: Dunmore
  readings:
    - r3
    - r4
```

	Reading	Station	Member	Features	Code
Baseline	+ Pressure	+ Latest weather, Temp C, F, Wind Bft, pressure	None	+ display latest weather for station	Zipped archive + Readme

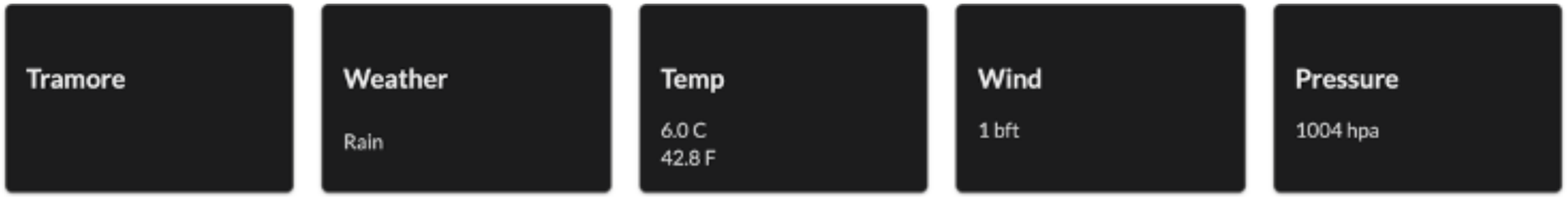


- Show latest conditions (last reading)

Weather conditions

Temp in both C & F

Wind in Beaufort



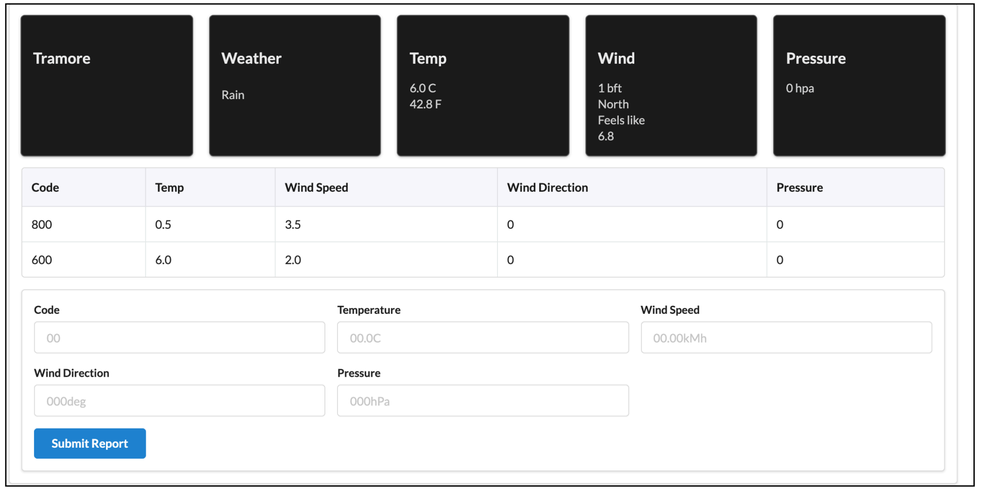
Station Name

Pressure

```
Reading(r1):
  code: 800
  temperature: 0.5
  windSpeed: 3.5
  pressure: 1001

Reading(r2):
  code: 600
  temperature: 6.0
  windSpeed: 2
  pressure: 1004

Reading(r3):
  code: 700
  temperature: 8.0
  windSpeed: 1
  pressure: 1000

Reading(r4):
  code: 200
  temperature: 0.5
  windSpeed: 3.5
  pressure: 999

Station(s1):
  name: Tramore
  readings:
    - r1
    - r2

Station(s2):
  name: Dunmore
  readings:
    - r3
    - r4
```

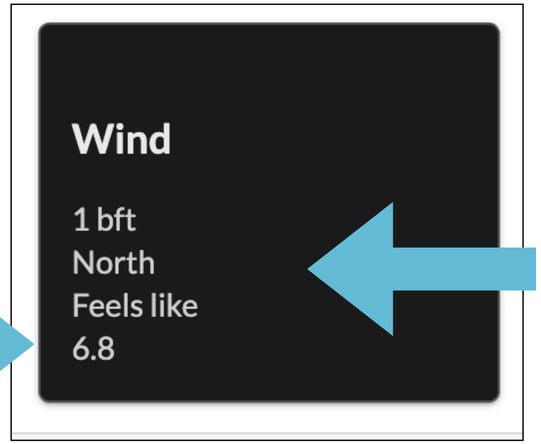
	Reading	Station	Member	Features	Code
Release 1	+ Wind Direction	+ Wind Chill, Wind Compass	None	Dashboard shows station list + button to open station view. Include forms to add new Station + new reading	Github repo

Add Station



Add Reading

Wind Chill



Wind Compass

```
Reading(r1):
  code: 800
  temperature: 0.5
  windSpeed: 3.5
  windDirection: 220
  pressure: 1001

Reading(r2):
  code: 600
  temperature: 6.0
  windSpeed: 2
  windDirection: 200
  pressure: 1004

Reading(r3):
  code: 700
  temperature: 8.0
  windSpeed: 1
  windDirection: 90
  pressure: 1000

Reading(r4):
  code: 200
  temperature: 0.5
  windSpeed: 3.5
  windDirection: 120
  pressure: 999

Station(s1):
  name: Tramore
  readings:
    - r1
    - r2

Station(s2):
  name: Dunmore
  readings:
    - r3
    - r4
```

	Reading	Station	Member	Features	Code
Starter	Code Temp Wind Speed	Station Name	None	Load and display stations + their readings from Yaml file	Zipped archive
Baseline	+ Pressure	+ Latest weather, Temp C, F, Wind Bft, pressure	None	+ display latest weather for station	Zipped archive + Readme
Release 1	+ Wind Direction	+ Wind Chill, Wind Compass	None	Dashboard shows station list + button to open station view. Include forms to add new Station + new reading	Github repo

Where to Start?

Starter

Consider adapting Playlist-2 application (Lab-09a) - refactoring the Song model to become an Reading & Playlist to become Station

Lab-09a Playlist-2

Move the playlist model into the database. Prime the database from a YAML file.

Baseline

Explore how to pass additional information to a view - not just a single object

Replicate how Playlist 3 moves the songs to their own view and move Readings to their own controller/view

Lab-09b Playlist-3

Move the playlists into their own view. Introduce routes for opening the playlist an...

Release 1

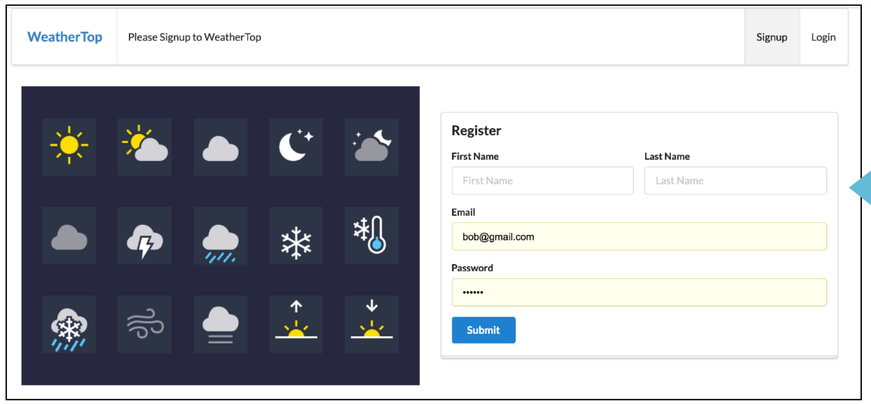
Copy the forms for creating playlists/songs from Playlist 4 - use these as the model for creating Readings/Stations

Lab-10b Playlist-4

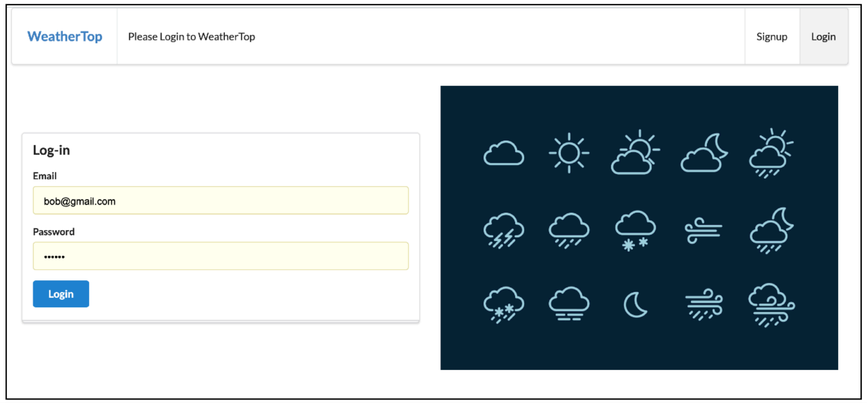
Introduce forms into a play application to enable the user to create playlists.

	Reading	Station	Member	Features	Code
Release 2		+Lat, Lng Max/Min (Temp, Wind, Pressure)	First Name, Last Name, Email, Password + Stations	Members can signup/log in. Members may create any number of weather stations. Members + sample stations + readings loaded from YAML	Deployed + Github repo + history

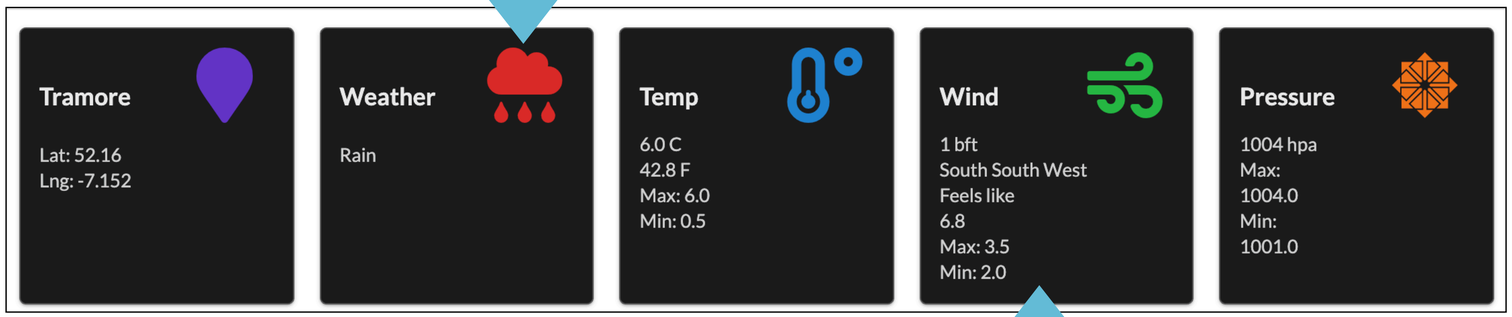
```
Readings as before…

Station(s1):
  name: Tramore
  lat: 52.160
  lng: -7.152
  readings:
    - r1
    - r2

Station(s2):
  name: Dunmore
  lat: 52.149
  lng: -6.994
  readings:
    - r3
    - r4

Member(m1):
  email: homer@simpson.com
  password: secret
  firstname: Homer
  lastname : Simpson
  stations :
    - s1
    - s2
```

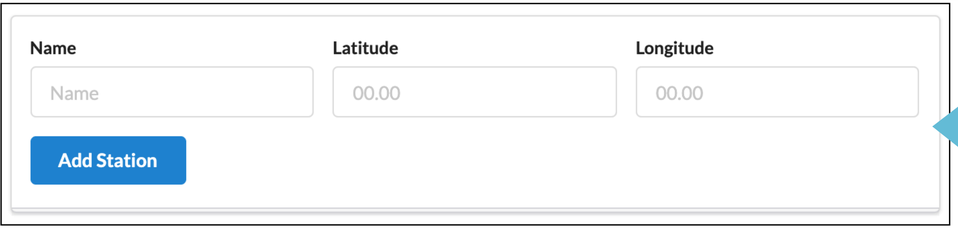


User can Sign up / log in



Suitable Icons





Add Station with Location

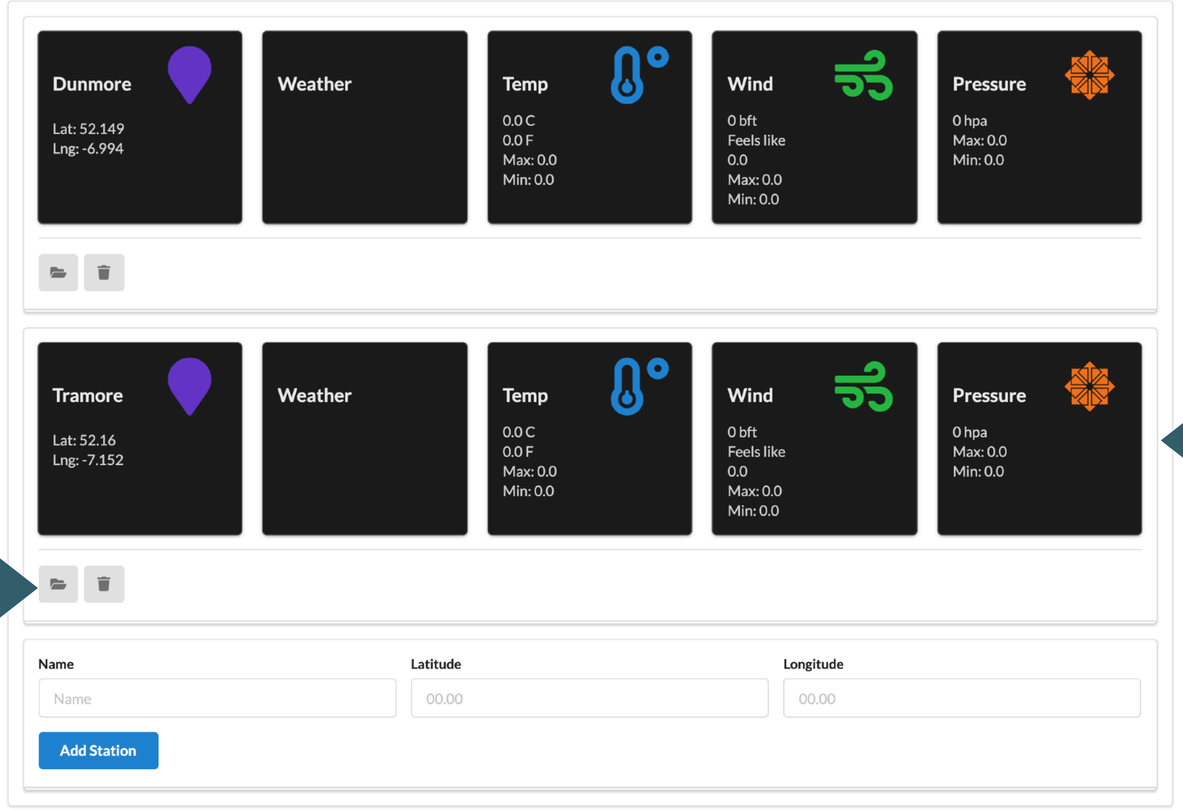
Max/Min Temp/Wind/Pressure

	Reading	Station	Member	Features	Code
Release 3	+ Time / Date	Temp, Wind Pressure Trends	User can edit their personal details.	Member dashboard list summary lists latest conditions for all stations. (alphabetically). Members can delete reports or stations	Deployed + Github repo + history tags)

```
Reading(r1):
  date: 2021-01-19 08:31:00
  code: 800
  temperature: 0.5
  windSpeed: 3.5
  windDirection: 220
  pressure: 1001

Reading(r2):
  date: 2021-01-20 09:34:00
  code: 600
  temperature: 6.0
  windSpeed: 2
  windDirection: 200
  pressure: 1004

Reading(r3):
  date: 2021-01-20 10:31:00
  code: 700
  temperature: 8.0
  windSpeed: 1
  windDirection: 90
  pressure: 1000

//...
```

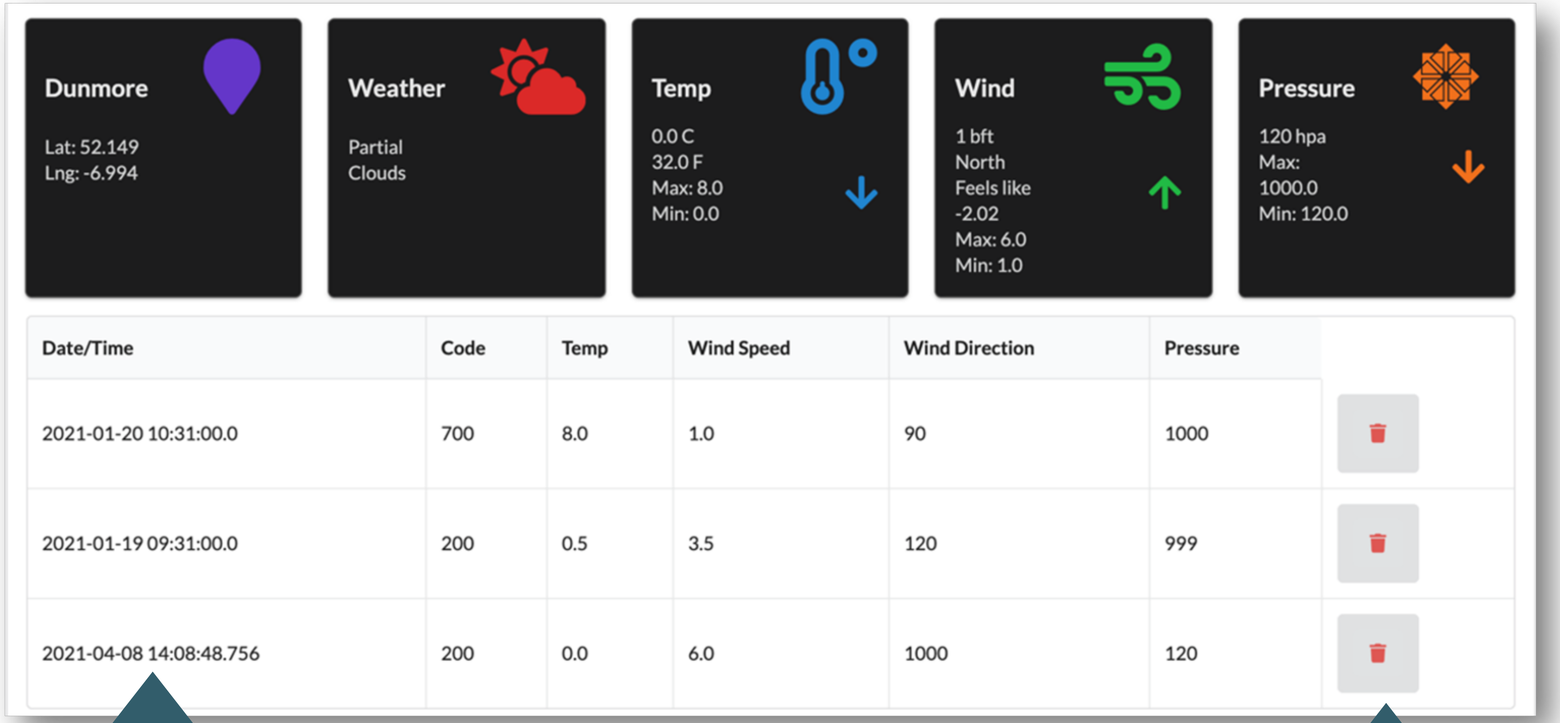
Stations listed alphabetically

Delete Station



Dashboard lists latest weather for all Stations (but not individual readings)

	Reading	Station	Member	Features	Code
Release 3	+ Time / Date	Temp, Wind Pressure Trends	User can edit their personal details.	Member dashboard list summary lists latest conditions for all stations. (alphabetically). Members can delete reports or stations	Deployed + Github repo + history tags)



Date/Time when reading is created

Delete reading

Trends

- Rising: Three most recent readings rising
- Falling: Three most recent readings falling
- Steady : neither of the above

Where to Start?

Release 2

Copy and adapt the Accounts Class + views from Playlist 5

Lab-11b Playlist 5

Incorporate Sessions into the latest Playlist Application

Implement delete strategy for Readings & Stations, adapting from approach implement in Playlist 3

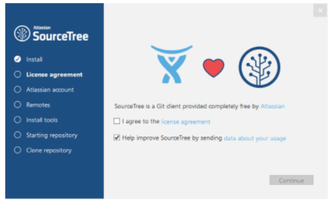
Lab-09b Playlist-3

Move the playlists into their own view. Introduce routes for opening the playlist an...

Release 3

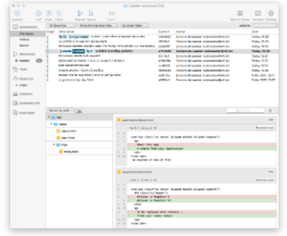
Advanced: complete git/deploy labs. Replicate approaches in WeatherTop

Lab-12a Git Setup



Install and Configure Git + the Sourcetree git GUI

Lab-12b Git Workflow

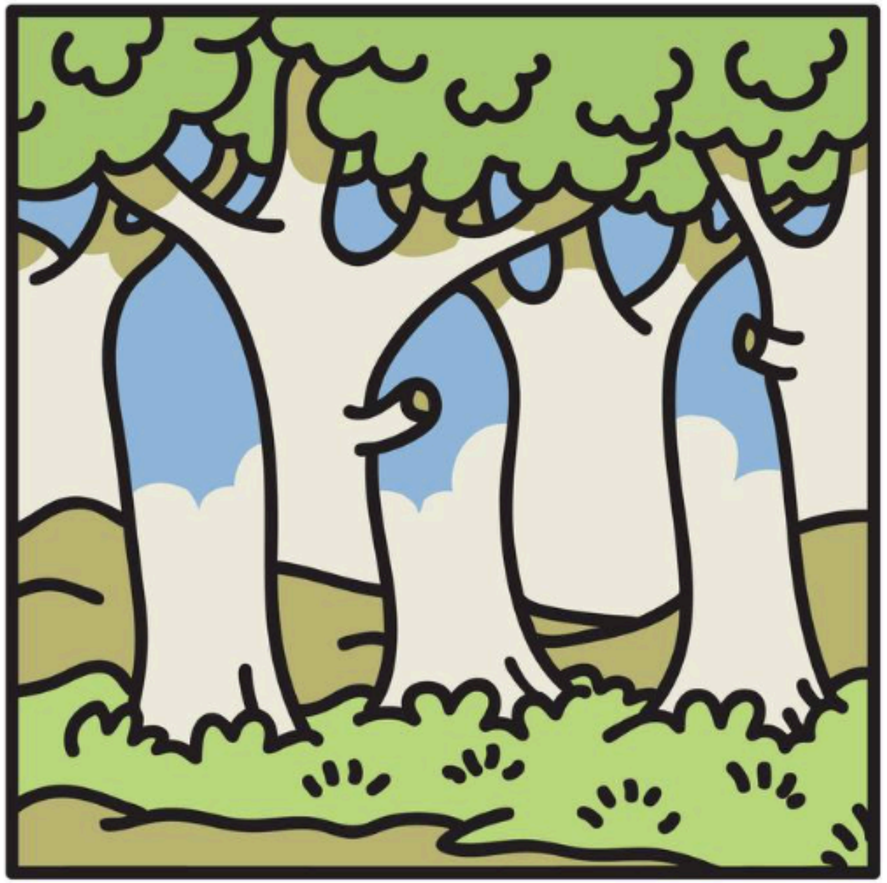


Rebuild Todolist - this time committing to git version control as the app is...

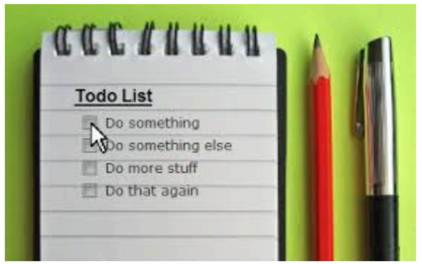
Lab-12c Deploy

Deploy a Play Application to the cloud

Difficulty Seeing the Wood from the Trees?

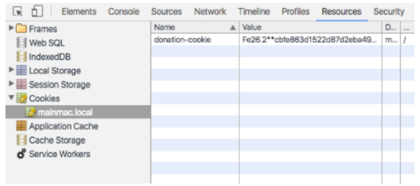


Lab-10a Todolist-1



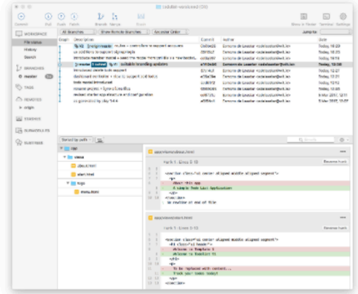
Develop a completely new application, using the techniques we have...

Lab-11aTodolist-2



Incorporate sessions tracking into the todo app

Lab-12b Git Workflow



Rebuild Todolist - this time committing to git version control as the app is...

These standalone labs tour the major concepts with a simplified application

(Optional)

	Reading	Station	Member	Features	Code
Starter	Code Temp Wind Speed	Station Name	None	Load and display stations + their readings from Yaml file	Zipped archive
Baseline	+ Pressure	+ Latest weather, Temp C, F, Wind Bft, pressure	None	+ display latest weather for station	Zipped archive + Readme
Release 1	+ Wind Direction	+ Wind Chill, Wind Compass	None	Dashboard shows station list + button to open station view. Include forms to add new Station + new reading	Github repo
Release 2		+Lat, Lng Max/Min (Temp, Wind, Pressure)	First Name, Last Name, Email, Password + Stations	Members can signup/log in. Members may create any number of weather stations. Members + sample stations + readings loaded from YAML	Deployed + Github repo + history
Release 3	+ Time/Date	Temp, Wind Pressure Trends	User can edit their personal details.	Member dashboard list summary lists latest conditions for all stations. (alphabetically). Members can delete reports or stations	Deployed + Github repo + history tags)

Algorithm References

(i) Weather Codes

100	Clear
200	Partial clouds
300	Cloudy
400	Light Showers
500	Heavy Showers
600	Rain
700	Snow
800	Thunder

(ii) Celsius to Fahrenheit Conversion

$$T_{(°F)} = T_{(°C)} \times 9/5 + 32$$

(iii) kM/hr to Beaufort Conversion

Beaufort	Beaufort Label	Km/h
0	Calm	1
1	Light Air	1-5
2	Light Breeze	6-11
3	Gentle Breeze	12-19
4	Moderate Breeze	20-28
5	Fresh Breeze	29-38
6	strong Breeze	39-49
7	Near Gale	50-61
8	Gale	62-74
9	Severe Gale	75-88
10	Strong storm	89-102
11	Violent Storm	103-117

(iv) Wind Direction Compass

Compass Direction	Degree Range
N	348.75 - 11.25
NNE	11.25 - 33.75
NE	33.75 - 56.25
ENE	56.25 - 78.75
E	78.75 - 101.25
ESE	101.25 - 123.75
SE	123.75 - 146.25
SSE	146.25 - 168.75
S	168.75 - 191.25
SSW	191.25 - 213.75
SW	213.75 - 236.25
WSW	236.25 - 258.75
W	258.75 - 281.25
WNW	281.25 - 303.75
NW	303.75 - 326.25
NNW	326.25 - 348.75

(v) Wind Chill Calculator

Wind chill = 13.12 + 0.6215 **T** – 11.37 (**V**^0.16) + 0.3965 **T** (**V**^0.16)

- **T** = Temperature in degrees Celsius
- **V** = Wind velocity in kilometers per hour

APPENDIX C: Q & A with Weather Top Inc.

Q: Is this the complete specification?

A: No, we are currently developing additional feature concepts – we will release these in another week or so.

Q: Will you accept partial implementations of a version if we run out of time?

A: Yes

Q: Must we complete all features specified in a version before attempting the next version?

A: No, we are happy to accept partial implementations of each version.

Q: How will you know which features we have implemented?

A: We will run all the versions. In addition, we ask that you submit a rubric/reflection grid indicating your achievements + a short demo video

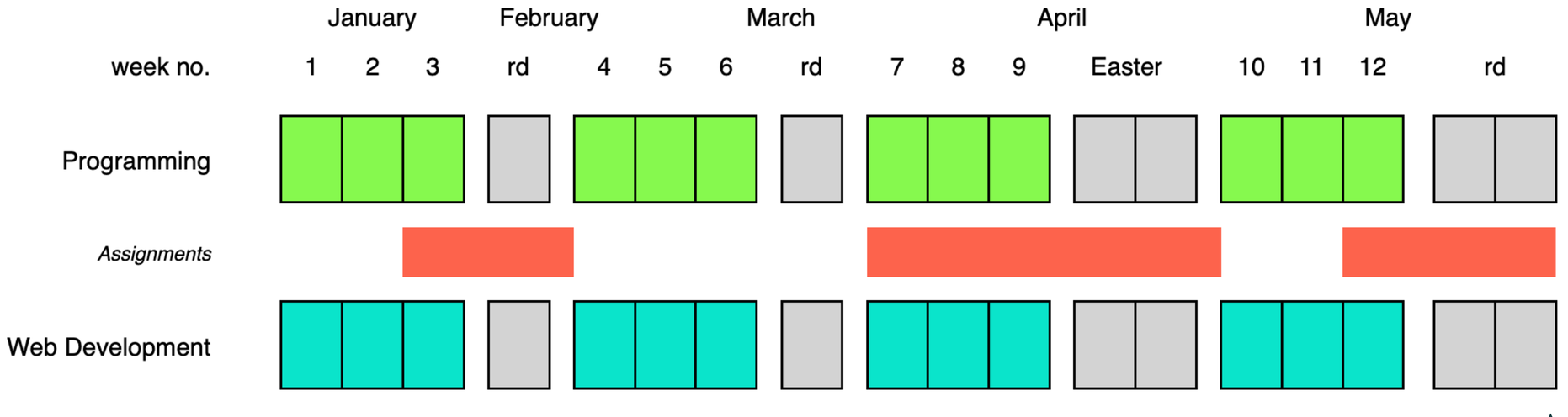
Q: Can we attempt any innovations over and above the specification?

A: Yes, but this only makes sense if you have completed the entire specification.

Q: Should we follow the 'Design Ideas' below precisely?

A: Not necessarily - if you have other ideas, we are happy to see them and they may be rewarded with additional payments.

Schedule



Calendar 2021

Semester 1		S	M	T	W	T	F	S	Modules
January	Week	3	4	5	6	7	8	9	
		10	11	12	13	14	15	16	Onsite
	1	17	18	19	20	21	22	23	programming & web dev
	2	24	25	26	27	28	29	30	programming & web dev
February	*3*	31	1	2	3	4	5	6	programming & web dev
	reading-week	7	8	9	10	11	12	13	
	4	14	15	16	17	18	19	20	programming & web dev
	5	21	22	23	24	25	26	27	programming & web dev
March	*6*	28	1	2	3	4	5	6	programming & web dev
	reading-week	7	8	9	10	11	12	13	
	7	14	15	16	17	18	19	20	programming & web dev
	8	21	22	23	24	25	26	27	programming & web dev
April	9	28	28	30	31	1	2	3	programming & web dev
	Reading week	4	5	6	7	8	9	10	
	Reading week	11	12	13	14	15	16	17	programming & web dev
	10	18	19	20	21	22	23	24	programming & web dev
	11	25	26	27	28	29	30	1	programming & web dev
May	12	2	3	4	5	6	7	8	
	reading-week	9	10	11	12	13	14	15	
	reading-week	16	17	18	19	20	21	22	

Due Date → Sunday 23rd May

Demos & Walkthrough June
24th May - 3rd June

Student Name : _____
Github Url : _____
Deployed URL : _____
Demo Video : _____

Grade Band	Reading	Station	Member	Features	Code
Starter					
Baseline					
Release 1					
Release 2					
Release 3					

Assignment Tips & Tricks

Share & Share Alike

Move Slowly and Don’t Break Things

Keep Model Classes Simple

Principle of Least Astonishment

Consider introducing “Utility” Classes

Share & Share Alike

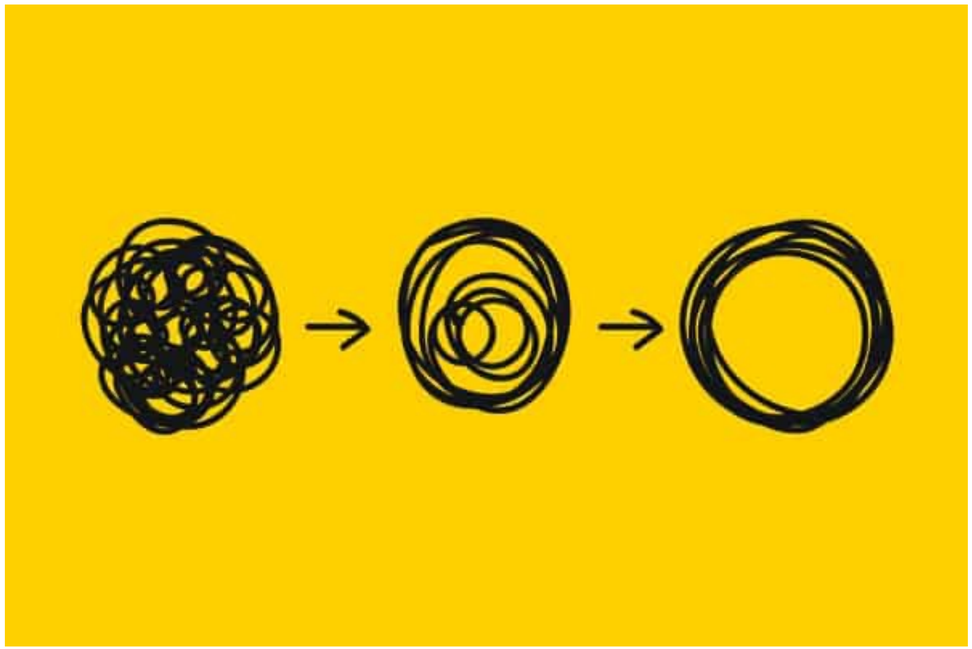
- Feel free to use any of the course projects as a foundation for your project.
- Feel free to experiment with snippets of code generously shared on #weather-top by other students
- If you do use a code snippet you don’t understand -
  - develop a theory as to why it works
  - seek clarification/confirmation either on #weather-top or via DM

Move Slowly and Don’t Break Things

- Move your project forward in small increments:
  - Make an adjustment
  - Stop and start to the project
  - Verify that your changes have had the desired effect
- If a changes causes the project to fail, undo to get to a working state again
- Keep regular backups

Keep Model Classes Simple

- Keep your model classes simple:
  - Keep using public fields
  - Do not over complicate the model class with a range of additional methods
  - It is, however, OK to include additional fields in the model classes
  - For example, Station could have latest temp in C & F.



Principle of Least Astonishment

- Write methods as simple an obvious as possible
- Keep each method focussed on its task, and keep the task simple
- Keep each method as short as possible
- This may mean larger number of smaller methods, rather than smaller number of large methods
- Make documenting methods unnecessary - choice of method and variable names conveys meaning

Consider introducing “Utility” Classes

- Much of software development is an exercise in moving complexity around so that it can be managed & evolved in an orderly manner.
- Utility Class Pattern:
  - Group a set of methods associated with some category of tasks into a standalone class
  - Use this class to perform these tasks when needed

Utility Class Example

- Station class has **someValue** field
- UtilityXXX is a class to do a range of conversions
- Further ways of grouping could also be considered

```
import utils.UtilityXXX;

public class StationCtrl extends Controller {

  public static void someMethod(Long id) {

    Station station = Station.findById(id);
    Reading lastReading = // get the last reading
    // …
    station.someValue = UtilityXXX.convertToZZZ(lastReading.someField);
    // …
    render("station.html", station);
  }
  //...
}
```

```
package utils;

public class UtilityXXX {

  public static double convertToZZZ(double someValueX) {
    double convertedValue = 0;
    // do conversion
    return convertedValue;
  }

  // other conversions....
}
```

More Tips

- Always try to keep the app close to a 'working' state (i.e. no syntax errors in Java + no runtime errors when programme is running)

- Don't attempt to run an app which has a Java syntax error. This error must be corrected before any successful run

- If you make a change, and the app generates an error, reverse back to a working state. This means you may be starting/stopping the app frequently. You can use commenting to keep the code in place.

- Once a run time error occurs, pinpoint what might be causing the issue, compare your code with a relevant lab, and try again. Try to develop a 'theory' as to why the error is occurring.

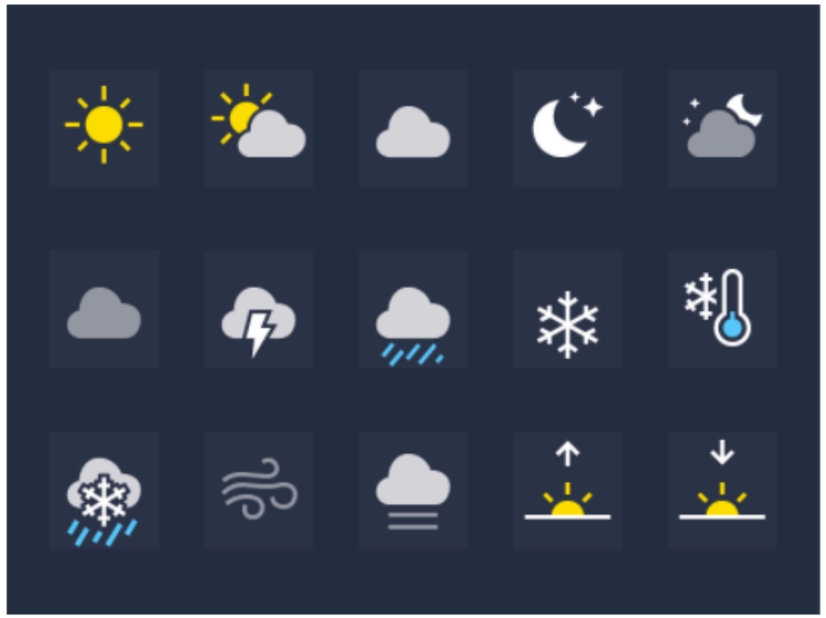
- Pay very close attention to spelling of routes, controller methods, + 'actions' in forms. All of these must be consistent

- Pay very close attention to plurality in spellings (station VS stations for instance), particularly in partials.

- If you do get stuck - and then get 'unstuck', and you think there is a valuable lesson - consider a short post summarising the experience (many useful post already!)

- Being in a 'stuck' state is often when much learning takes place as you try different tactics - so don't be too self critical.

Project 2 Specification



Specification for
Assignment 2