John Calandrino
Collaborative Systems

Collaborative Weather-Reporting Application Manual

This manual describes the operation of my collaborative weather reporting application, designed within the Sync framework. My weather application, called WeatherApp in the Sync framework, merges data about different weather conditions reported from different regions. Additionally, it combines multiple weather reports from the same region into a single report that merges all of the conditions.

To start a new synchronized WeatherApp, start the Sync framework and go to File→New→WeatherApp while the entry representing the server replica is highlighted. A window similar to Figure 1 should appear. Note that unless your resolution is 1200 pixels wide or higher, you will likely have to horizontally scroll to see the rest of the window.

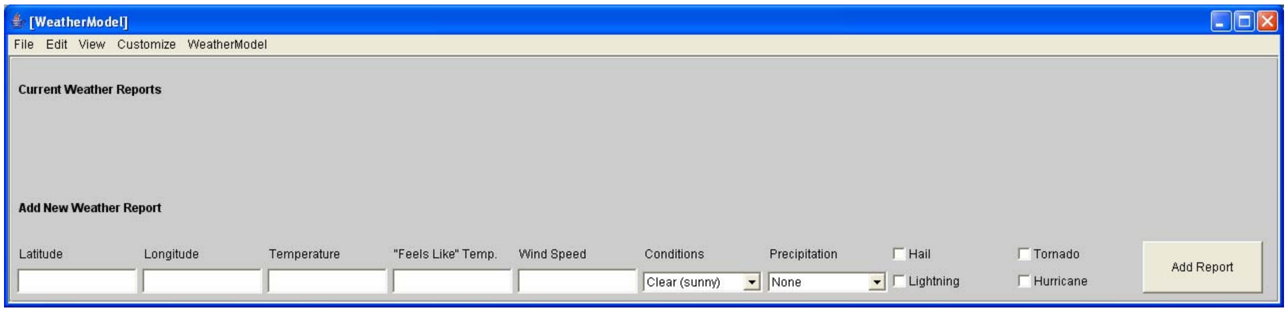
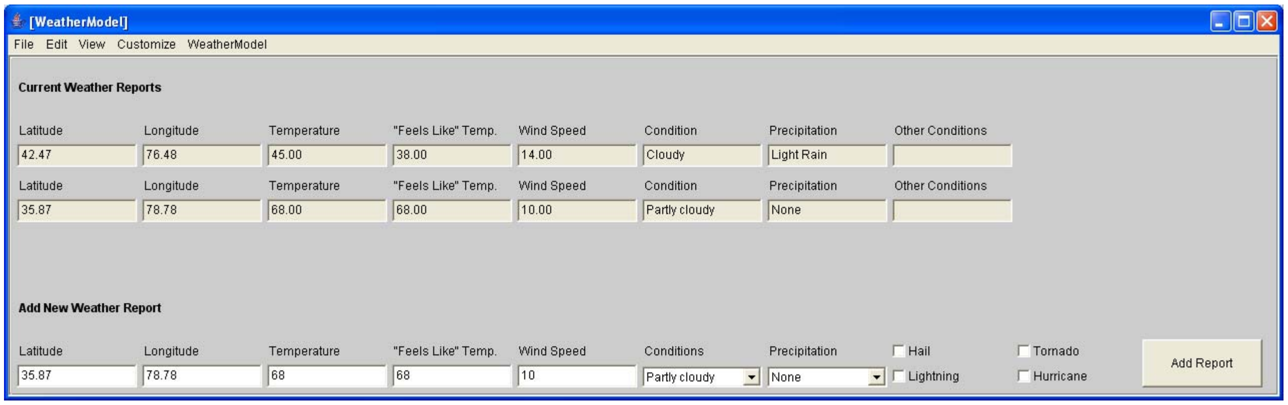


Figure 1: A New WeatherApp Window

There is a title "Current Weather Reports" under which reports are usually shown, but since the application starts with zero reports, there is currently only a grey area. At the bottom of the screen are a number of fields and choices followed by an "Add Report" button. This allows the user to add a new report to the system. Note that there are quite a number of components to a weather report, as outlined in Figure 2. Figure 3 shows the WeatherApp window after two weather entries have been added to the system, one for latitude 42.47 degrees, longitude 76.48 degrees, and one for latitude 35.87 degrees, longitude 78.78 degrees (all latitudes are degrees north and longitudes are degrees west - southern latitudes and eastern longitudes are represented by negative numbers).

Component	Description
Latitude	Latitude of reported weather conditions.
Longitude	Longitude of reported weather conditions.
Temperature	Current temperature.
"Feels Like" Temp.	Current perceived temperature (factoring in wind chill, heat index, etc.).
Wind Speed	Current wind speed.
Condition	Current general conditions, ranging from clear and sunny to cloudy to foggy.
Precipitation	Current precipitation, ranging from none to heavy rain, snow, or sleet.
Hail	Can check the box to report hail as part of current conditions.
Lightning	Can check the box to report lightning as part of current conditions.
Tornado	Can check the box to report a tornado as part of current conditions.
Hurricane	Can check the box to report a hurricane as part of current conditions.

Figure 2: Description of All Components of a Weather Report



Current Weather Reports

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Condition	Precipitation	Other Conditions
42.47	76.48	45.00	38.00	14.00	Cloudy	Light Rain	
35.87	78.78	68.00	68.00	10.00	Partly cloudy	None	

Add New Weather Report

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Conditions	Precipitation		
35.87	78.78	68	68	10	Partly cloudy	None	☐ Hail ☐ Lightning	☐ Tornado ☐ Hurricane

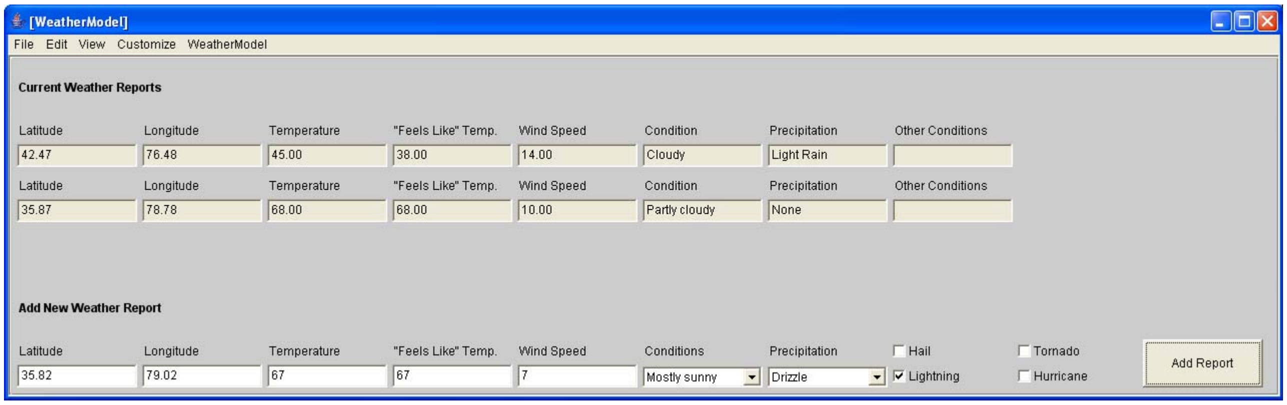
Figure 3: WeatherApp Window After Two Reports Added

Next, let's demonstrate the ability of WeatherApp to combine data from multiple weather reports. WeatherApp employs a merge algorithm for every component of a weather report, as outlined in Figure 4, whenever two weather reports are considered to be nearby in terms of latitude and longitude. The WeatherApp checks the list for pairs of nearby reports anytime the list is updated. Currently, if the latitude and longitude of two weather reports are *both* within a quarter degree of each other (which works out to approximately 15-20 miles of distance maximum), then we consider the weather reports close enough to merge. Note from Figure 4 that most of the merges simply take the numerical average of reported weather parameters, which works out great for naturally numeric fields, and for the condition as well, since increasing condition values represent "degrees of cloudiness" (both condition and precipitation fields are represented internally as a numeric integer value by the WeatherApp application). To merge precipitation conditions, a more complicated algorithm is used, based primarily on the fact that there are two major categories of precipitation, rain and snow, and this requires us to check the type of precipitation more closely before merging. To merge the checkbox conditions, we simply report a condition if either of the reports include the condition. Note that this means that a checkbox condition, once reported, will never disappear - this decision was made because these conditions tend to be severe, and it seems best to always report them. Perhaps the condition, precipitation, and checkbox fields could all be periodically cleared or reset, so that calm and less severe weather conditions could once again be accurately represented in the system. The frequency of such a reset would depend on how frequently clients were providing weather reports. Another method might involve including a time-to-live with weather reports. Again, the length of the TTL would depend on the frequency of client updates.

Component	Merge Algorithm for Two Reports R1 and R2
Latitude	Take the average of latitudes in R1 and R2.
Longitude	Take the average of longitudes in R1 and R2.
Temperature	Take the average of temperatures in R1 and R2.
"Feels Like" Temp.	Take the average of "feels like" temperatures in R1 and R2.
Wind Speed	Take the average of wind speeds in R1 and R2.
Condition	Take the average of conditions (represented numerically in the WeatherApp system) in R1 and R2. This works since conditions range from clear to foggy, thus representing increasing "degrees of cloudiness."
Precipitation	Precipitation must be merged according to the types of precipitation reported by R1 and R2. If either R1 or R2 reported no precipitation, then any reported precipitation makes it directly into the merged report. If the precipitation types are both rain or both snow, we can simply take the average across that type of precipitation. If one reports rain and the other snow, then they merge into sleet. The same goes for if one reports freezing rain and the other reports any other type of precipitation. Alternately, if one reports sleet and the other reports any type of precipitation, then the other type of precipitation is included in the merged report.
Hail	Hail is part of the merged report of either R1 or R2 reported hail (OR operation).
Lightning	Lightning is part of the merged report of either R1 or R2 reported hail (OR operation).
Tornado	A tornado is part of the merged report of either R1 or R2 reported hail (OR operation).
Hurricane	A hurricane is part of the merged report of either R1 or R2 reported hail (OR operation).

Figure 4: Merge Algorithm To Combine Data From Multiple Weather Reports

As an example of merging, let's submit the following weather report as shown at the bottom of the screen capture in Figure 5 (with the data from Figure 3 still in the system). Figure 6 shows the result. Notice that the same number of entries exist, but that the second report was merged with the new report as outlined in Figure 4. These algorithms are just examples of what WeatherApp could do; one could develop more sophisticated nearness and merging algorithms and plug them into the program with relatively little trouble.



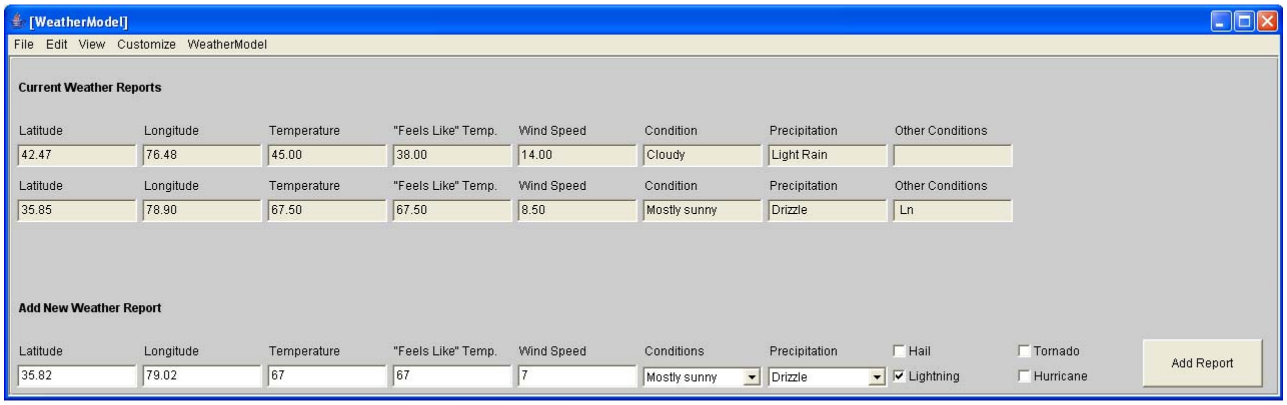
Current Weather Reports

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Condition	Precipitation	Other Conditions
42.47	76.48	45.00	38.00	14.00	Cloudy	Light Rain	
35.87	78.78	68.00	68.00	10.00	Partly cloudy	None	

Add New Weather Report

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Conditions	Precipitation		
35.82	79.02	67	67	7	Mostly sunny	Drizzle	☐ Hail ☑ Lightning	☐ Tornado ☐ Hurricane

Figure 5: Immediately Before Submitting New "Nearby" Weather Report



Current Weather Reports

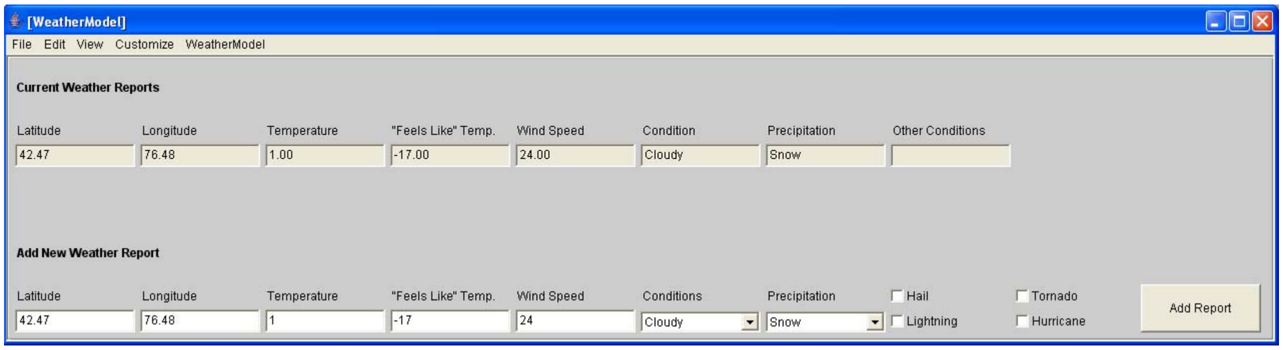
Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Condition	Precipitation	Other Conditions
42.47	76.48	45.00	38.00	14.00	Cloudy	Light Rain	
35.85	78.90	67.50	67.50	8.50	Mostly sunny	Drizzle	Ln

Add New Weather Report

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Conditions	Precipitation		
35.82	79.02	67	67	7	Mostly sunny	Drizzle	☐ Hail ☑ Lightning	☐ Tornado ☐ Hurricane

Figure 6: Merge Result from Submitting New "Nearby" Weather Report

Finally, as this is a collaborative application, let's demonstrate the synchronization properties of WeatherApp across Sync clients. WeatherApp should merge weather reports from both clients into one common set of reports. As an example, start the WeatherApp on one client ("Client 1") and add a single report to it, as shown in Figure 7. Next, start another client ("Client 2") and immediately synchronize between clients so that Client 2 sees the same WeatherApp. Now, we have the same identical set of reports on both clients, as shown in Figure 8. Next, Clients 1 and 2 each add a new weather report that is not nearby to either the current report in the system, or the report being added by the other client. Client 2 also adds an additional weather report that *is* nearby to the weather report added by Client 1. (Note that "real-time" synchronization is not enabled during this example.) Figure 9 shows the results of all of these adds. Finally, Figure 10 shows what both clients see after synchronization, assuming synchronization works correctly (in the current version of Sync, it might take multiple "Synchronize" calls from both clients to get things fully and correctly synchronized). Note that the first report remains unchanged, the independent report provided by Client 2 is now seen by both clients, and the nearby reports of Clients 1 and 2 were combined into a single report. Currently, the example does not work as well for real-time synchronization - it appears that sometimes the merge occurs more than once and the values of components in merged reports are somewhat incorrect.



Current Weather Reports

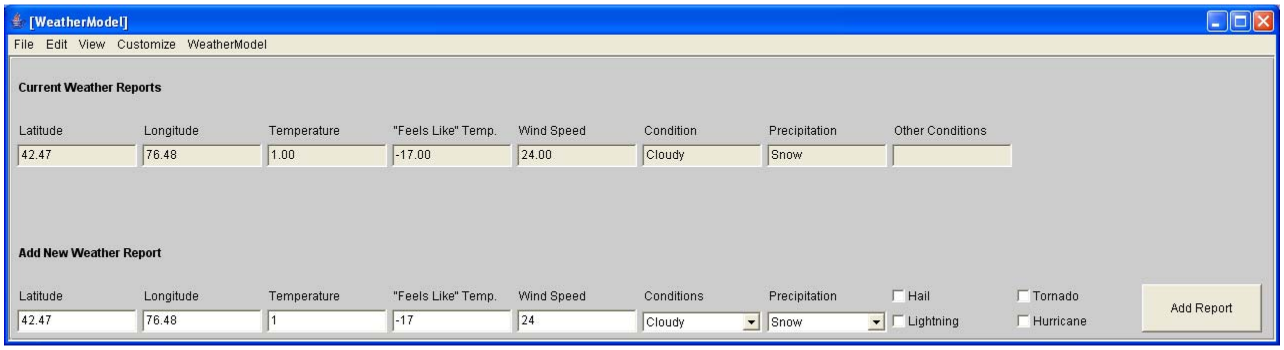
Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Condition	Precipitation	Other Conditions
42.47	76.48	1.00	-17.00	24.00	Cloudy	Snow	

Add New Weather Report

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Conditions	Precipitation		
42.47	76.48	1	-17	24	Cloudy	Snow	☐ Hail ☐ Lightning	☐ Tornado ☐ Hurricane

Figure 7: WeatherApp Window with One Weather Report

Client 1



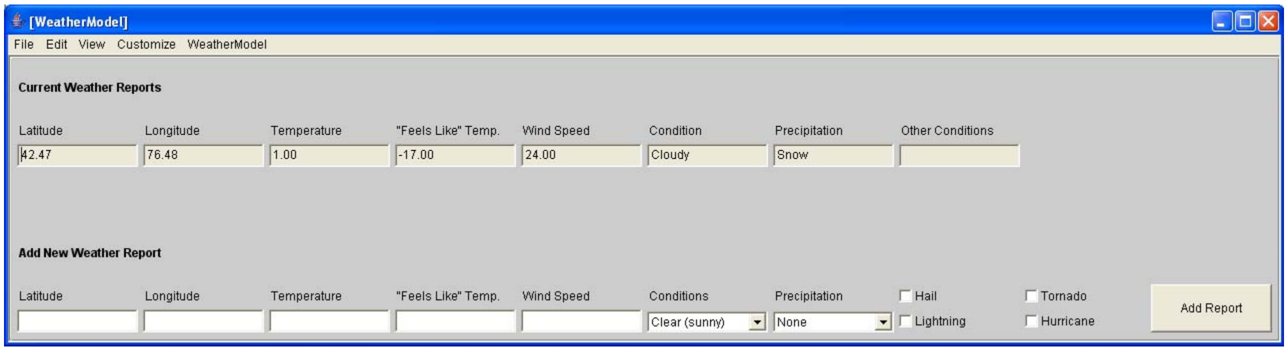
Current Weather Reports

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Condition	Precipitation	Other Conditions
42.47	76.48	1.00	-17.00	24.00	Cloudy	Snow	

Add New Weather Report

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Conditions	Precipitation		
42.47	76.48	1	-17	24	Cloudy	Snow	☐ Hail ☐ Lightning	☐ Tornado ☐ Hurricane

Client 2



Current Weather Reports

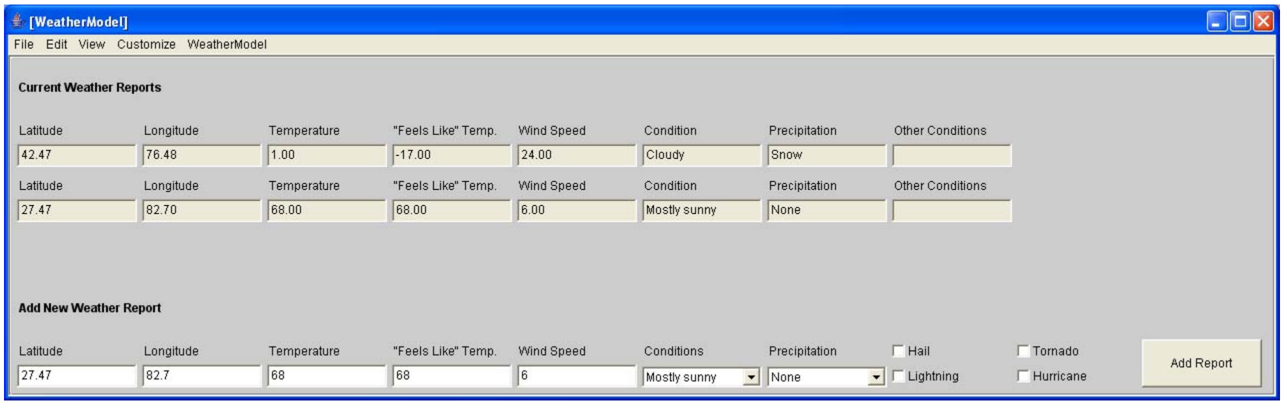
Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Condition	Precipitation	Other Conditions
42.47	76.48	1.00	-17.00	24.00	Cloudy	Snow	

Add New Weather Report

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Conditions	Precipitation		
					Clear (sunny)	None	☐ Hail ☐ Lightning	☐ Tornado ☐ Hurricane

Figure 8: WeatherApp Windows of Both Clients After Synchronization

Client 1



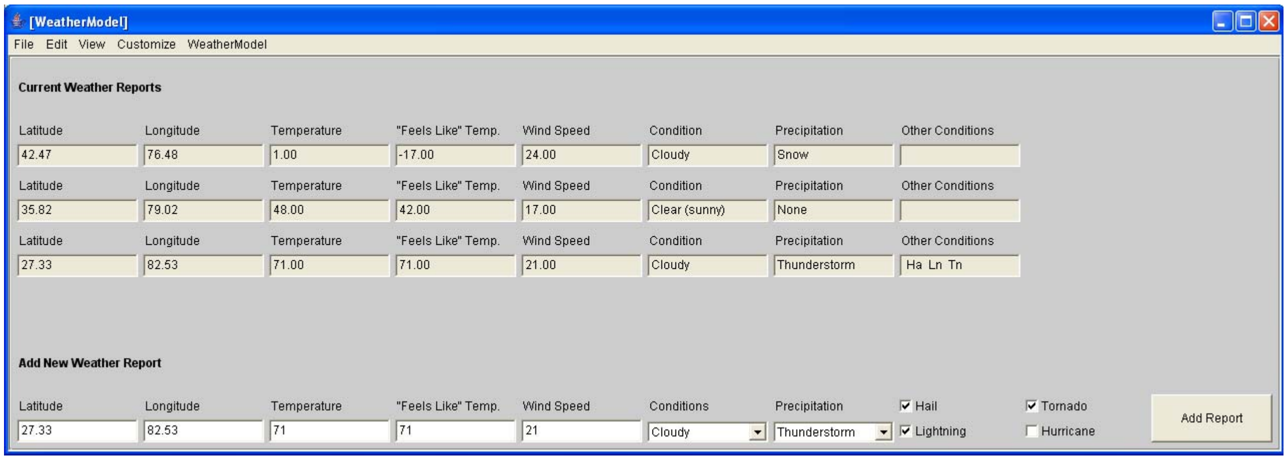
Current Weather Reports

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Condition	Precipitation	Other Conditions
42.47	76.48	1.00	-17.00	24.00	Cloudy	Snow	
27.47	82.70	68.00	68.00	6.00	Mostly sunny	None	

Add New Weather Report

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Conditions	Precipitation		
27.47	82.7	68	68	6	Mostly sunny	None	☐ Hail ☐ Lightning	☐ Tornado ☐ Hurricane

Client 2



Current Weather Reports

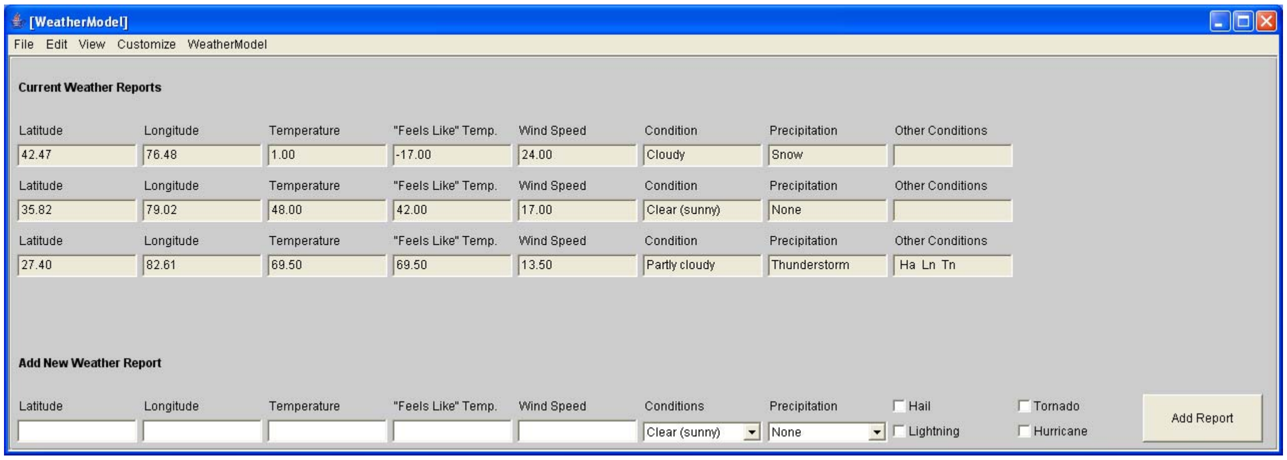
Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Condition	Precipitation	Other Conditions
42.47	76.48	1.00	-17.00	24.00	Cloudy	Snow	
35.82	79.02	48.00	42.00	17.00	Clear (sunny)	None	
27.33	82.53	71.00	71.00	21.00	Cloudy	Thunderstorm	Ha Ln Tn

Add New Weather Report

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Conditions	Precipitation		
27.33	82.53	71	71	21	Cloudy	Thunderstorm	☑ Hail ☑ Lightning	☑ Tornado ☐ Hurricane

Figure 9: WeatherApp Windows of Both Clients After Adding Several New Reports

Client 1



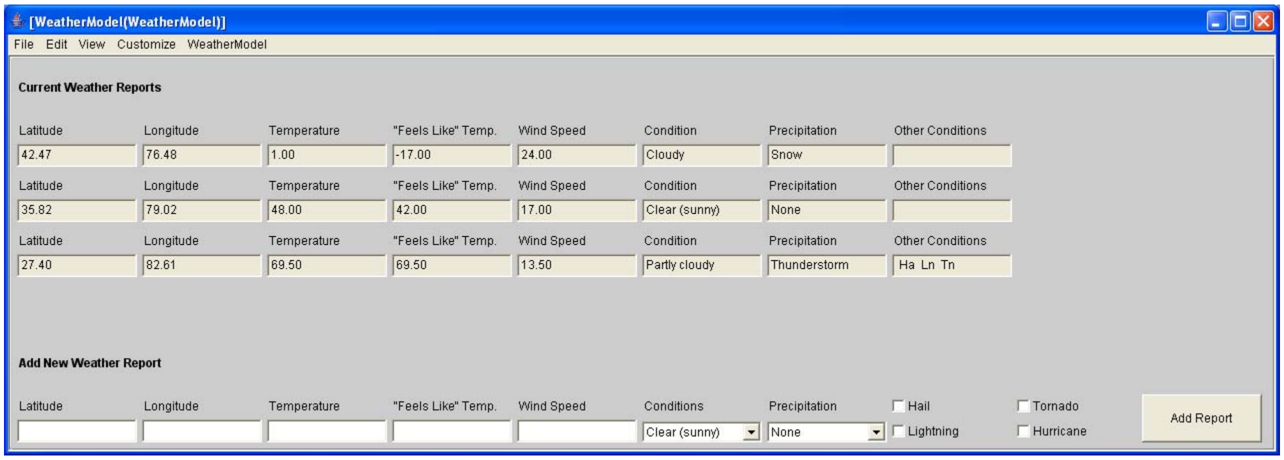
Current Weather Reports

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Condition	Precipitation	Other Conditions
42.47	76.48	1.00	-17.00	24.00	Cloudy	Snow	
35.82	79.02	48.00	42.00	17.00	Clear (sunny)	None	
27.40	82.61	69.50	69.50	13.50	Partly cloudy	Thunderstorm	Ha Ln Tn

Add New Weather Report

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Conditions	Precipitation		
					Clear (sunny)	None	☐ Hail ☐ Lightning	☐ Tornado ☐ Hurricane

Client 2


[WeatherModel(WeatherModel)]
File Edit View Customize WeatherModel

Current Weather Reports

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Condition	Precipitation	Other Conditions
42.47	76.48	1.00	-17.00	24.00	Cloudy	Snow	
35.82	79.02	48.00	42.00	17.00	Clear (sunny)	None	
27.40	82.61	69.50	69.50	13.50	Partly cloudy	Thunderstorm	Ha Ln Tn

Add New Weather Report

Latitude	Longitude	Temperature	"Feels Like" Temp.	Wind Speed	Conditions	Precipitation		
					Clear (sunny)	None	☐ Hail ☐ Lightning	☐ Tornado ☐ Hurricane

Figure 10: WeatherApp Windows of Both Clients After Synchronization of Reports in Figure 9

As we can see, WeatherApp is a collaborative application that allows users (or fully-automated weather stations, etc.) to report weather conditions, and allows other clients to see those weather reports. It also merges nearby reports into a single report, thus simplifying the data and allowing the client to see a summary of conditions rather than having to combine reports on their own. The merge and nearness algorithms employed seem to work reasonably well, though they could certainly be upgraded. Upgrading these algorithms, providing a map in the GUI, and fixing the issues with "real-time" synchronization (assuming they are WeatherApp problems and not Sync problems) are three immediate areas of future work. Additionally, it might be useful to provide an interface for other types of clients, such as fully-automated weather stations, to make automatic reports to the WeatherApp system.

Appendix A: Programming Interfaces of Sync Objects Representing Replicated Model

WeatherModel - The replicated model, holds a Vector of WeatherStats objects (the list of weather reports).

```
public WeatherModel()
```
- Create a new WeatherModel object.

```
public void addElement(WeatherStats element)
```
- Add a WeatherStats object to the vector (add a weather report).

```
public void removeElement(WeatherStats element)
```
- Remove a WeatherStats object from the vector (remove a weather report).

```
public int numReports()
```
- Returns the number of reports in the vector (i.e. the vector size).

```
public Enumeration elements()
```
- Returns an Enumeration for the vector of weather reports.

```
public ListIterator listIterator()
```
- Returns a ListIterator for the vector of weather reports. Used during view refresh when removing/merging "nearby" entries.

```
public void signalElementRemoved(WeatherStats element)
```
- Called after an element is removed after a merge during a view refresh.

```
public void setElementAt(WeatherStats element, int index)
```
- Set the value of an element in the vector of weather reports.

```
public void addVectorMethodsListener(VectorMethodsListener vml)
public void removeVectorMethodsListener(VectorMethodsListener vml)
```
- Add or remove vector change listeners.

```
public void addVectorListener(VectorListener vl)
public void removeVectorListener(VectorListener vl)
```
- Add or remove vector listeners, used for automatic user interface generation. Since a custom object view was defined, these may no longer be needed but are kept in case they are needed again later.

WeatherStats - Contains data for an individual weather report.

```
public WeatherStats()
```
- Create a new WeatherStats object.

```
public void mergeData(WeatherStats toMerge)
```
- Merge WeatherStats object data into this object.

These methods get and set attributes of the weather report data.

```
public float getLatitude()
public void setLatitude(float newVal)

public float getLongitude()
public void setLongitude(float newVal)

public float getTemperature()
public void setTemperature(float newVal)

public float getFeelsLikeTemp()
public void setFeelsLikeTemp(float newVal)

public float getWindspeed()
public void setWindspeed(float newVal)

public int getCondition()
public void setCondition(int newVal)

public int getPrecip()
public void setPrecip(int newVal)

public boolean getHail()
public void setHail(boolean newVal)

public boolean getLightning()
public void setLightning(boolean newVal)

public boolean getTornado()
public void setTornado(boolean newVal)

public boolean getHurricane()
public void setHurricane(boolean newVal)


public void addPropertyChangeListener(PropertyChangeListener l)
public void removePropertyChangeListener(PropertyChangeListener l)
```

- Add or remove property change listeners.