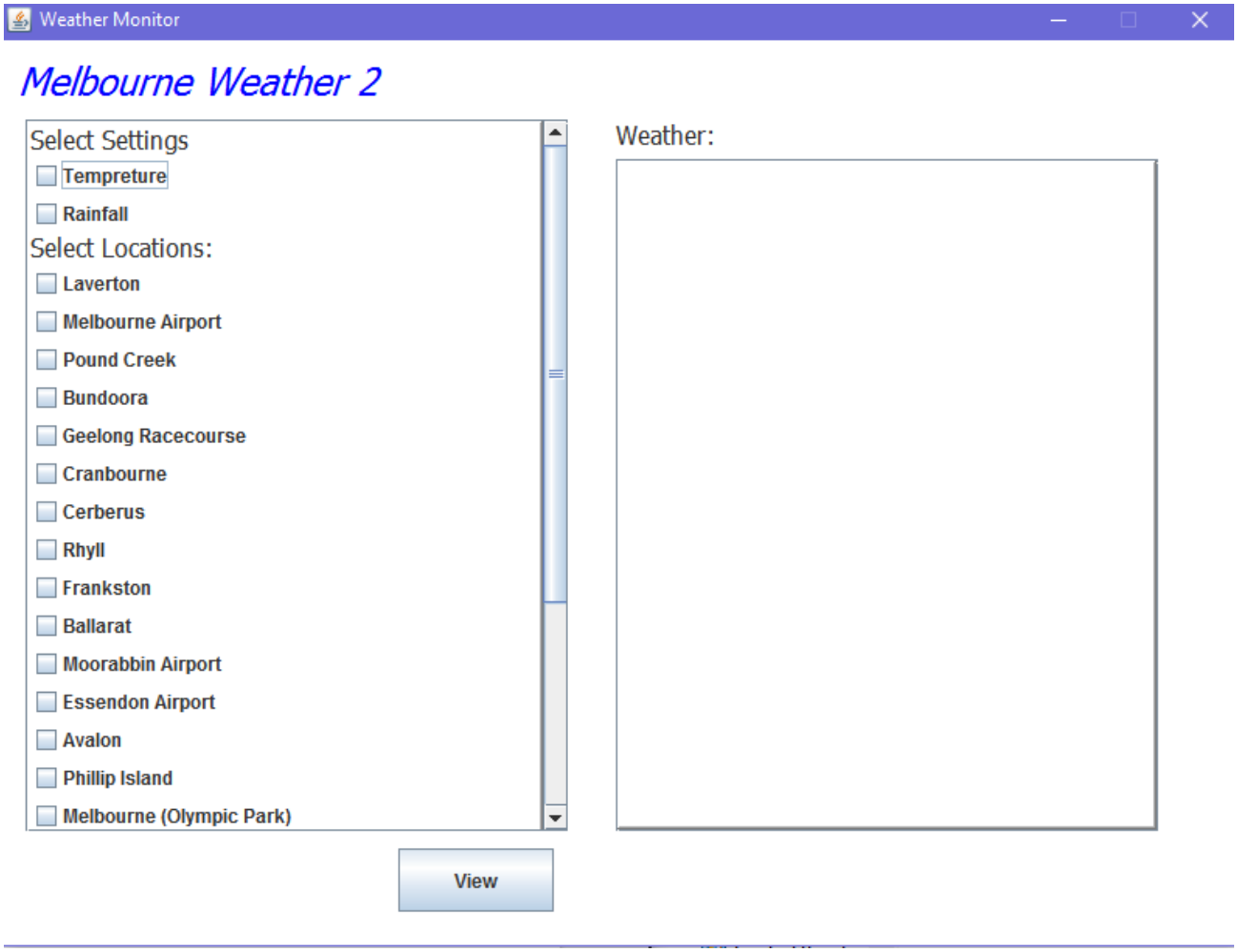
Natasha Van der Walt Gomezgani Phiri

DESIGN DOCUMENTATION

The Melbourne Weather 2 service is developed using the *Java EE* programming language and *Apache* Tomcat 6 and Axis2. The program is shown below:



1.Data Design

The Melbourne weather 2 service program is designed using the responsibility- driven approach. This uses a *SOAP client-server* model of design. Both the client and server are classes or instances of classes where at a time, they both represent an object. These sides commit to exchanging information as a system to form the service.

The responsibilities in this design are for the client to perform the private task of receiving the data, while the server is supposed to know the information as the public responsibility. Hand in hand they work to distribute data.

2. Architecture Design

The overall scope of the software architecture is the ***level 2 encapsulation***. This is because most classes and parts of their interfaces are not visible. This is evident in the design as the Development view is missing. The existing *views* of the architecture's design consist of the:

a. **Logical view** that shows the functionality the system. The *state diagram* and *class diagrams* contain this information as they show the way different components of the system behave.

b. **The Process view** which shows the dynamics and communication within the system. The *Activity diagram* and *sequence diagram* highlight this as they show the way processes and objects are exchanged between classes.

Scenarios are also included using the *use cases*. This acts as the third level of view to show how users interact with the system.

The design does not go further than level 4 as classes in code were not grouped into the higher-level structures such as packages at this stage.

The *architectural style* of this application uses the client-server model as said recently. Because of this, it is possible to expand our view of design to satisfy the general reusability solution.

3. Interface Design

The screenshot of the program is shown. It is evident that the program follows good design principles suitable to usability as it explicitly fulfills the user requirements without drawing too much attention to itself.

The interface also follows the design principle of reuse and maintenance as it shows plenty of room for upgrades such as addition of more settings, more locations are more ways to display the information in the provided frame. This is because of the swing components used that add the extra layer of abstraction between the user and the system in the model which adds flexibility with high cohesion with the controllers and low coupling among views.