Testing the Broadcast Receiver, called the AlarmReceiver, was one of the first steps necessary in developing the product. The first test was using MainActivity to create a PendingIntent that was sent to the AlarmManager. One minute later, android would send an intent to the broadcast receiver, which would then take the data and create a notification to be displayed on the action bar. As the project progressed, the test required for the AlarmReceiver had to expand. Such as a check to make sure the intent had a notification tied to it, so that the information needed push the notification to the action bar was possible. Other tests as well, such as making sure the notification was still enabled, and that the system could handle when more than one intent was sent to it at once. The tests for the AlarmReceiver still need to be expanded. There are no tests in place to check what happens when the user changes the time of a notification after the PendingIntent is sent to the AlarmManager.

Creating the three AbstractBaseEvent objects, Lifestyles, Reminders, and Notification objects, also required its own tests to verify that they worked correctly. The original testing process was to create objects in code and place values in to make sure that everything worked properly and placing values in the data structures did not crash the app. Later, more tests were put into place that tested the relationship between the objects. More tests that could be included would be the creation of scripts that created a large amount of all three data types and made sure the relationship between them worked correctly, if there were many more objects.

Testing the GSON library with the three AbstractBaseEvent objects was another type of testing that had to be done. GSON was selected in the first place because when testing JSON, it showed that JSON was inadequate to perform the tasks that were needed of it. There is no need to perform further testing with GSON as the testing performed so far, converting the AbstractBaseEvents to and from GSON, has proven that it works properly for our needs.
Another part of the project that required a large amount of testing is storing all the data structures in a way that would be accessible, even if the app is off. The primary way that storage has been tested so far is taking the AbstractBaseEvent objects and calling the storage functions. We found the best way to test storage was to use storage, passing the objects created during the AbstractBaseEvent testing and then calling storage to see if the values came back out. This way has worked perfectly so far, and the only problems encountered with storage was confusion on how it worked, rather than whether or not it did work. Testing in this way will continue until the end, as there has been no better method provided and has worked beautifully up to this point.

Another part of the project that had to be tested is the use of the Array Adapters to display the necessary information. Each of the three type of AbstractBaseEvents required its own Array Adapter unique from the other types. The testing for this was also part of the testing for the object creation and storage. After the objects were created in code, they would then call storage to store the data, and then the Array Adapter would call on storage for these objects and then display them

to the user. Which helped test other aspects of the project to make sure they worked right.
The final step in testing would be crowd-based testing. Part of giving the application to users would be to see if the application does not break when they use it. The other part, and at this point more important part, would be to make sure the User Interface is usable to someone who was not involved in the development of the application. Fortunately, the way the User Interface is designed is simple enough that making changes to it will not be costly, in terms of time or changing the underlying structure of how the system works, because the underlying system was designed to work indepently of how the UI works.