# Requirements and Design – Revel Software Engineer Practical Exercise

To separate the narrative of the practical exercise description from the salient points related to project requirements, I’ve copied out the paragraphs immediately below and created a new document. Design features appear later in the paper.

## Requirements

From the distributed document comes the following:

Now you are ready to get down to business! In talking with your Scrum team and your technical lead, you have determined that the first task you are going to take on is to implement a new feature that determines when to best contact individuals based on the weather. Acme’s product team has determined the following:

* The best time to engage a customer via a text message is when it is sunny and warmer than 75 degrees Fahrenheit
* The best time to engage a customer via email is when it is between 55 and 75 degrees Fahrenheit
* The best time to engage a customer via a phone call is when it is less than 55 degrees or when it is raining.

You will need to build a function that uses the API provided by openweathermap.org at the following url: <http://api.openweathermap.org/data/2.5/forecast?q=minneapolis,us&units=imperial&APPID=09110e603c1d5c272f94f64305c09436>.

The documentation for this API can be found here: <https://openweathermap.org/forecast5>.

Your function should determine what outreach method is best for Minneapolis, MN over the next 5 days. The forecast will give you data for multiple points during a given day, you may choose to use that data how you wish. In general Acme uses technologies from the Java family to implement services and has a preference for single page JavaScript applications for the frontend using Angular or React. However, you may use whatever language you would prefer. The function can be run as either a web application or a command line application. If it is a web application it should display a 5 day calendar with the best outreach method listed for that date, if it is a command line application it should print out each date and the best outreach method for that day on a new line.

## Design

The watchwords for this design are simplicity and standardization. The requirements section above allows for the use of my current favorite language, Java, so I plan to use it. The UI may be graphical or command line; in the interest of simplicity, I plan to run my application from the command line. Based on that choice, the application output will be text showing the preferred method for contacting customers per day over the next five days.

The data returned by the OpenWeather resource is more granular than needed by the application being created, *i.e.,* it gives eight data points per day while we need only one. Since the choice of which data to use for selecting the contact method has been left to the application writer, the forecast element in the response with the **dt\_text** field containing 12:00:00 (noon) will be the only portion of each day’s data that will be used.

Since the interpretation of the forecast data has been left up to the application writer, and no default condition has been specified, the **temp\_max** field will be used for all temperature values; if no such field is found, or if found contains no valid data, the temperature defaults to 0F. In the same vein, the **weather** field will be searched for the word “rain” to determine whether the sky is rainy; all other sky conditions (*e.g.,* clouds, snow) are considered to be sunny for the purposes of categorization.

## Details

It was discovered that the example JSON file used for the unit tests was of a slightly different format than the fresh OpenWeather response. This fact led to some hair-tearing initially but resulted in code that is a bit more generous in what it accepts and so is capable of handling either format correctly.

The code makes extensive use of the Apache StringUtils library to manipulate the data extracted from the JSON elements of the OpenWeather response.

The code for the project has been added to a GitHub repository created locally and available globally. The code appears as a zipped file in the public vernrein/RePrEx repository in GitHub within the master branch. It is accompanied by a ReadMe file and a copy of this file.