

# Software Requirements Specification

## For Weather App

**Instructor:**  
**Team Members:**  
**Cycle:**  
**Date Submitted:**

Document template copyright 2005-2015, CCI Faculty. Version 2.3. Use permitted under Creative Commons license CC-BY-NC-SA. See <http://creativecommons.org/licenses/by-nc-sa/3.0/>.

**Commented [Instr1]:** Please use this report template as provided. Do not change template elements such as section numbers, headings or stock text, reorganize the report, or delete sections.

You may add sections at the end or add sub-sections if needed. You should also adjust page breaks so major headings are placed appropriately in the final document.

Items in angle brackets, < >, are parts of the report that you must replace with appropriate information. All the angle brackets should be gone in your final submission.

All comments (like this one) should be DELETED before submitting the final version of the report. Comments in a draft version are fine.

This document is patterned after a subset of IEEE Std 830-1998 and you should consult that document for additional information about what each section should contain.

Be particularly aware of these key points:

The SRS is a key document for communicating with the client. The document should be written with a client audience in mind. Sections 1 and 2 are particularly oriented toward general readers.

“The SRS writer(s) should avoid placing either design or project requirements in the SRS” (IEEE 830, 4.1). Be particularly careful to avoid making design decisions here. If you do make a design decision, it should be in the “Design Constraints” section.

## Grading Rubric - Requirements Specification

This rubric outlines the grading criteria for this document. Note that the criteria represent a plan for grading. Change is possible, especially given the dynamic nature of this course. Any change will be applied consistently for the entire class.

Achievement	Minimal	Exemplary	Pts	Score
<b>Content (80)</b>	Section(s) missing, not useful, inconsistent, or wrong.	Provides all relevant information correctly and with appropriate detail		
Introduction Scope Definitions			10	
User Profile			20	
Functional Requirements			30	
Performance & Design Requirements			10	
Data Requirements			10	
<b>Writing (20)</b>				
Grammar and Spelling	Many serious mistakes in grammar or spelling	Grammar, punctuation, and spelling all correct	10	
Expression	Hard to follow or poor word choices	Clear and concise. A pleasure to read	5	
Tone	Tone not appropriate for technical writing	Tone is consistently professional		
Organization	Information difficult to locate	All information is easy to find and important points stand out	5	
Layout	Layout is inconsistent, visually distracting, or hinders use	Layout is attractive, consistent, and helps guide the reader		
<b>Late Submission</b>			-10 -25	
<b>Total</b>			100	

# 1 Introduction

This section provides an overview of <system name>. Scope provides a short description of the product and how it is useful; Definitions explains terms with which a reader may not be familiar; and User Profile identifies the ways different groups of people would make use of <system name>.

My team and I plan on making a weather app specifically for people in college who want to know what they should wear before leaving their dorm. Walking outside feeling out the temperature, walking back in to change is a very lengthy process if you live in a college dorm or anywhere in an apartment building. Our product will take the current and upcoming weather and send you notifications based on when you're schedule. We want our app to have as little user interaction with the application as possible.

## 1.1 Scope

We're trying to create a weather app that integrates the college lifestyle with the weather. It'll recommend what clothes you should wear, it'll tell you whether to bring an umbrella or sunglass outside and do all of this with as little interaction with our app as possible. Our goal is to have our clientele go through their day as normally as possible, but they will get notifications from our app. We plan on doing this by gaining access to the Google API.

## 1.2 Definitions, Acronyms, and Abbreviations

There are no definitions, acronyms, or abbreviations at this time.

## 1.3 User Profile

Ideally, we're looking for college students, at least to start. This app could be useful for commuters as well, since they must travel far for their work and would like to be prepared for the weather. Those are our intended users; however, this application could be used by anyone that finds it useful. We all find that people constantly ask others what the weather is like outside and what they should be wearing, we believe our app could help remedy this situation and fill a hole in the market.

# 2 External Interfaces

Our application must interact with an external weather API and google calendar. We must use google calendar to sync up their schedules and make it interact with the weather API, but we can have the interactions take place on the user's device. The only output device we'll need is the user's screen.

**Commented [Instr2]:** See IEEE 830 section 5.1 for a discussion of this section and its subsections.

**Commented [Instr3]:** Provide a short description of your product written for a potential client. What does your product do? Why would the client want to use it?

**Commented [Instr4]:** If you have no material for this section, simply say so, e.g., "There are no definitions, acronyms, or abbreviations at this time." Do not define common technical terms. Do spell out all acronyms that you use.

**Commented [Instr5]:** Identify groups of people who will use the product and describe how each would use it. Try to identify groups that will use the product in different ways.

Name each group and use these names wherever possible throughout the product documents, and especially in the detailed requirements. If one group can perform all the functions of another group, explain this here. Try to choose names that are meaningful to people who would buy or use the product.

For example, for a sales tracking system, the groups might include sales rep, sales manager, financial analyst, general manager, and customer. Your description would explain what functions the product would provide for each group, and might note that sales managers can perform all the functions of a sales rep, plus more (and you would define what that "more" is).

**Commented [Instr6]:** This section identifies ways in which your system interacts with people and other systems. Note that you are not defining the details of this interaction, but just identifying flows of information into and out of your system. Most of the detailed definition will happen in the functional requirements and in your design phase.

## 2.1 User Interface

The user will use the application through a user interface. The interface should be very simple only needing a way for them to set up what notifications they want, a sign in for their google calendar, and a tab so that they can see the current weather.

## 2.2 Data Interface

Our system will collect data from a weather API, it'll take the current weather and future weather, if the API permits it, for the user's area. It'll then take that information as well as the user's calendar data and compare the two. We will need a list of all of the events on that person's calendar so that we know when they are leaving the house and can give recommendations according to when they need it.

# 3 Specific Requirements

## 3.1 Functional Requirements

The statements below define the functional requirements for the system.

### Find Weather API - <The Search>

We have been searching for an API that we can afford, aka free, and it is very difficult since most of them require some type of payment. We have found some free trials, but it won't give us the future weather reports which is the whole point to our application.

### Learn to use weather API - <Research and Development 1>

Finding the weather API is the easy part, once we find it, we actually have to learn how to retrieve and use the data that we need. This part shouldn't be too difficult because we are all somewhat experienced, but it's certainly time consuming.

### Get the Google Calendar API and learn how to use it - <Research and Development 2>

We are using the google calendar for our testing phase since it is used by many users across the US and the world. It syncs very well through user's systems and is our best option during our testing phase of development.

### Data interaction - <Creating a Relationship>

The weather and google calendar API's need to interact with each other, instead of doing this directly, we plan on taking the data from each and putting it in the app. Having everything interact on the local system may increase performance requirements slightly, but it is the best option for development time and makes it easier to design the system to our liking.

### Publication - <The Gathering>

Once we have a working prototype, we have to compile everything into an apk file and see if it works nicely on the android system. This is a relatively easy, yet time consuming portion of the project especially if we run into design or functionality problems.

**Commented [Instr7]:** Briefly describe whether and how your system interacts with people.

For example:

Whizbang will interact with users via a user interface. There will be separate sections of this interface for buyers, sellers, and administrators.

Do NOT design the interface in this section or talk about technology to implement it. Save that for your design document.

**Commented [Instr8]:** Briefly identify any ways that your system interacts with other systems by exchanging data. Note that this is not about your system's own database, but about data that crosses your system boundary.

For example:

myWeather will obtain current weather information from the National Weather Center and long term climate data from NOAA.

**Commented [Instr9]:** This is the heart of the document and typically the largest section. Use subsections as needed, and organize the requirements in a useful way.

Individual requirements can be quite detailed. Each entry within a requirement should be as long as needed.

Avoid generic references to the "user." Wherever possible refer by name to the user groups defined in the User Profile section.

**Commented [Instr10]:** Define a unique ID and short name for each requirement. The IDs may be sequential or be structured to reflect the organization of the requirements. The name should describe the function.

For example:

1. Place a bid

## Artistic Design - <Touch Up>

We want this app to look nice yet very minimalistic, since the application is based on notifications. Making sure the app isn't overbearing is key since the less interaction the user has with the app the better. If we could have the user set up the app and it work from there on out.

## 3.2 Performance Requirements

The statements below define the performance requirements for the system.

### Internet - <User>

The user must have access to the internet so that we can update the weather and access their calendar through Google's API.

### Phone- <User>

The phone must be an android smartphone, we would try to develop for an iphone since the user base in the united states would be much bigger, but prototyping on an android is much easier and doesn't require licensing.

### Weather API - <N/A>

We need to be able to access the weather reports. This is will be accessed through a free weather API from an unknown provider as of right now.

### Google API - <Google>

We need to access Google's calendar API so that we have access to the users calendar information. This and the weather API are the reasons we need the user to have access to the internet.

## 3.3 Design Constraints

### 3.3.1 Constraint: Getting a free weather API

Reason: Getting a weather API with our low budget of \$0 is proving to be quite difficult. We have found a couple, but they aren't giving us future weather reports, only the weather at that exact date and time. This simply will not do for both testing and functional reasons.

### Constraint: Only runs on android

Reason: Even though our target audience is College students, who, for the most part, carry apple devices, we need to develop for android devices. It is much easier to develop on an android device and doesn't require any licensing for development.

## 3.4 Data Requirements

<ID> - <Name>

Name	Type	Size	Comment
Calendar	Date/Event	User	We need access to the user's calendar

**Commented [Instr11]:** Specify performance requirements in this section

Do not invent performance requirements that have no basis in fact. For example, if you specify a required response time you should be able to justify the threshold value you choose. On the other hand, do identify performance requirements that are real even if you don't know the correct threshold value. Simply note that a value needs to be determined (since you can't test without one!).

For example: eBay responses for site navigation requests will complete in 1 second or less. (Note: the threshold value is approximate and will be refined later.)

**Commented [Instr12]:** Describe any constraints that will limit your design and explain the reason for the constraint.

Example:

Constraint: Whizbang will be an android phone app.

Reason: The client has specified that a mobile app is a requirement and android runs on the most devices. Other mobile platforms may be addressed later.

**Commented [Instr13]:** Define the data required by the system. Think of this section as part of your conversation with the client, and use this section to capture the client's view of the data. Ideally, you would like a client to review what you write.

This section should include lists of data elements organized into data entities that a client would recognize. You may also want to tag data elements that have unique values for each instance of an entity (and so might become a key in the database).

This section will provide the basis for creating a database design in your design specification, but **you should not design the database here**. The entities here should be based on the client view of the data. They may or may not become database tables.

For example, if your system has invoices, the client would think of an invoice as a single entity, so you might use one table here. In the design phase, this will probably become multiple database tables, e.g., one for general invoice information and a separate table for the line items of the invoice.

A good way to find data requirements is to read the functional requirements, pick out all the nouns, and think about how that noun will be represented by data in the system.

Repeat the table as needed

**Commented [Instr14]:** Record data element names that the client knows. For example: "customer" not "cust"; "first name" not "fname". Write in the client language, not names you might use internally in a program.

**Commented [Instr15]:** Data type. These can be generic types such as Number, String, Date

**Commented [Instr16]:** Maximum length. Required for character string data elements; optional for others

**Commented [Instr17]:** Explain what the data element is unless it is clear from the name

		Defined	information in order for this application to exist
Weather	Date/Time	Data based defined	We need to gather weather information from a database and push it to the application