**Software Requirements Specification**

**for**

**CI Event Coordinator**

**Version 1.0 approved**

**Prepared by <author>**

**California State University Channel Islands**

**10/5/15**

**Table of Contents**

**Table of Contents**

**Revision History**

**1.** **Introduction**

1.1 Purpose

1.2 References

**2.** **Overall Description**

2.1 User Classes and Characteristics

2.2 Operating Environment

2.3 Design and Implementation Constraints

2.4 Assumptions and Dependencies

**3.** **External Interface Requirements**

3.1 User Interfaces

3.2 Hardware Interfaces

3.3 Software Interfaces

3.4 Communications Interfaces

**4.** **System Use Cases**

4.1 Use case name and identifier

4.2 Withdraw money from ATM (U2)

4.3 Deposit money into ATM (U3)

**5.** **Other Nonfunctional Requirements**

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

**6.** **Other Requirements**

**7.** **System Requirements Chart**

**Appendix A: Analysis Models**

**Appendix B: To Be Determined List**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Nicolas Fredrickson, Jonathan Burtson, Timothy Holcome | 10/5/15 | First version of Requirement Doc | 1.0 |

# **Introduction**

## **Purpose**

This is a group project for COMP 350 CIEvents, web-based application is required where users are able to create and manage events in various locations such as CSUCI main campus, Santa Barbara campus, Santa Rosa Island etc., and invite others. Info about time and place of the event should be provided on line. The system should alert the event creator X days before the event day in case of weather condition change and suggests the closest day with suitable weather. All event participants should be notified in case of weather forecast change via email. Invited users should be able to decline or accept the invitation. Model, View Controller pattern should be utilized in implementation.

## **References**

None!

# **Overall Description**

## **User Classes and Characteristics**

As this is a basic project, we mostly expect the users to be fellow computer science students, instructors, and enthusiasts. As such, we will expect a basic (if not advanced) familiarity with the computer and websites.

## **Operating Environment**

This project will come in the form of a web-based app, hosted on Heroku. We will be targeting what we consider the main three web browsers - Google Chrome, Mozilla Firefox, and Apple’s Safari. As for supported versions, we are aiming for versions supporting certain CSS3 features, including animations, unique web-fonts, and rounded boxes.

## **Design and Implementation Constraints**

While this project is meant to be a companion app on myCI, we will not have access to the various school resources that entails. If this were truly a project sponsored by CI, we would have access to already existing databases, in the form of user data. As it is, we will have to dynamically create this data as the app is used. Having the deployment run by CI would also make the app easier to scale, as we would not have to pay for additional “dynos”.

## The **Assumptions and Dependencies**

Our most major dependency, in terms of the software, will be the weather API we choose. We are definitely making the assumption that the API will even be able to provide us with the weather data that we want. We want to get weather information that is in depth enough to be able to make the distinction between bad weather and good weather in a particular location and time. We’re also assuming that weather data will be accurate - after all, weather is typically held up as a paramount example of instability and unpredictability. We also will need to be able to use the API to determine the weather of a particular location, as determined by user input. If this is not an easy thing for the API we may need to determine alternatives.

# **External Interface Requirements**

**SKIP ALL OF SECTION 3 for the SRS… We’ll do this as part of the GUI Prototype**

## **User Interfaces**

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

*Think about this in terms of standards, not specific features. For example,*

* *all buttons will have a black border*
* *all fonts will be Arial*
* *Draw a screen template showing “main area”, “menu here”, “status bar” and describe each component. If you have multiple screen layouts depending on the user’s current task/settings, describe them*
* *What screen resolutions will you support?*
* *Will you be Section 508 compliant? Are there any other standards you support?*
* *etc…*

## **Hardware Interfaces**

*<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>*

*If you system doesn’t include hardware, then you’ll have none. If it has hardware components, then you should describe (at a high level) how you interface with that hardware.*

## **Software Interfaces**

*<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>*

*These are internal connections to things like databases, web servers. You mainly need to explain that you have them, but (for CS421) I don’t expect detailed information about how you actually connect to them and use them. Just explain that you* ***do*** *connect to them and use them for storage of customer information, or to process incoming web requests, etc… These are internal components of your system.*

## **Communications Interfaces**

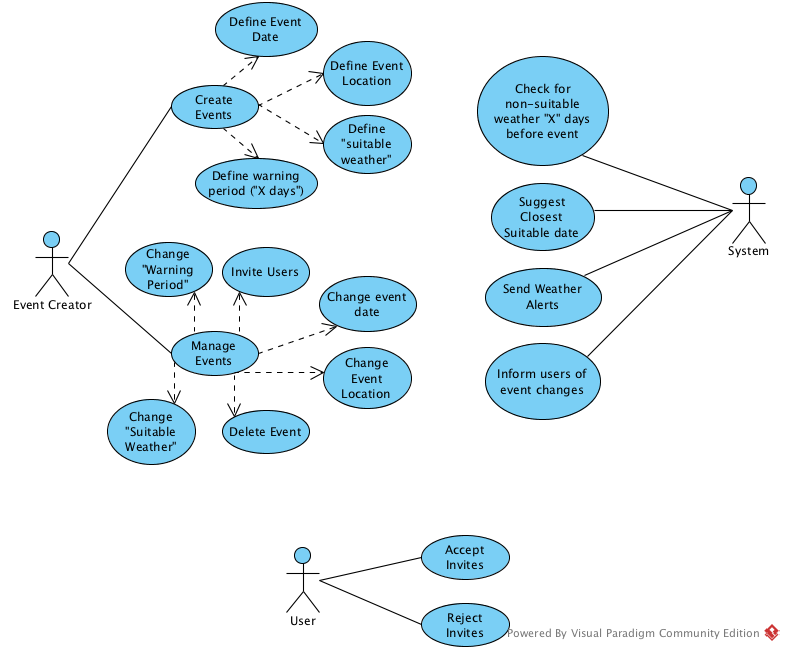
*<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>*

*These are external communication mechanisms. Do you connect to a bank computer to verify credit card information? That is NOT part of your system, so it is an external communication you have. Describe it here. Do you have other systems connecting in to yours to perform some function? That would also go here.*

# **System Use Cases**

*The overall use case diagram should be here.*

*The text description of each use case should follow.*



## Event Creation (EC1)

1. **EC1**
2. **Objective** – In this instance, the event creator is attempting to create and define new events.
3. **Priority** – High
4. **Source** – John Jacob Jingleheimer Schmidt (R & D)
5. **Actors** - Event Creator, System
6. **Flow of Events** 
   1. **Basic Flow**
      1. The event creator creates a new event - they trigger, in some way, the event creation process.
      2. The user defines the date of the event.
      3. The user defines the location of the event
      4. The user defines “suitable weather” for the event.
      5. The user defines the warning period - the X in “X days”
   2. **Exception Flow(s)**
      1. If any of the entered data is invalid, we do not accept the data and prompt the user for data again. The user should be informed of the errant data.
7. **Includes** - Immediately flows into EC2.
8. **Preconditions** - User must be logged in, in order to associate the event with that user.
9. **Post conditions** - An event exists.
10. **Notes/Issues** - None!

## Event Management (EC2)

1. **EC2**
2. **Objective** – The event creator will be able to modify the properties of the event
3. **Priority** – Medium
4. **Source** – Old Jimmy Jim Jamboree (R & D, Section 3)
5. **Actors** – Event Creator, System
6. **Flow of Events** 
   1. **Basic Flow**
      1. The user can invite other users.
      2. The user can change the warning period.
      3. The user can change the event date.
      4. The user can change the event location.
      5. The user can change the “suitable weather”
      6. The user can delete the event.
   2. **Exception Flow(s)**
      1. If any of the entered data is invalid, we do not accept the data and prompt the user for data again. The user should be informed of the errant data.
7. **Includes** - Uses SM to inform attendees of changes.
8. **Preconditions** - The user must be logged in. The event we are managing must exist.
9. **Post conditions** - An event exists.
10. **Notes/Issues** - Note that, in this use case, the user doesn’t flow from option to option; rather, the user is presented with options 16.1.1 through 16.1.6 simultaneously.

## User (U)

1. **U**
2. **Objective** - The user will wither accept or reject waiting invites.
3. **Priority** – Medium
4. **Source** – Ziggy Stardust (Some basement or something)
5. **Actors** - User, System
6. **Flow of Events** 
   1. **Basic Flow**
      1. The user views their invites.
      2. The user either accepts the invite, or rejects the invites.
   2. **Alternative Flow(s)**
      1. If the user has no invites, inform the user they have no friends.
7. **Includes** - Uses SM to inform the Event Creator User.
8. **Preconditions** - User is logged in, so we can retrieve their invites.
9. **Post conditions** - The user will be added to the event’s list of users to notify.
10. **Notes/Issues** - None.

## Daily Weather Checks

1. **U2**
2. **Objective** – A daily weather check occurs, check returned weather data against event bad weathers. If is a bad weather send alert to event creator, accepted invitees, and undecided invitees.
3. **Priority** – High
4. **Source** – Programming
5. **Actors** – Weather, Calender, User
6. **Flow of Events** 
   1. **Basic Flow**
      1. Event is created by user
      2. Weather is checked
      3. Event is confirmed
      4. Time passes
      5. Daily check occurs
      6. If returned weather does not conflict with bad weather selections repeat from 6.1.4
      7. Event occurs
      8. Event is deleted.
   2. **Alternative Flow 1** – At step 6.1.6 the amount is not a multiple of $20
      1. If there is a weather conflict send email alert to creator and accepted invitees and undecided invitees.
      2. Creator of event decides to reschedule or cancel
      3. Event is rescheduled return to step 6.1.2
   3. **Alternative Flow 2** – From 6.2.3
      1. Event is canceled
      2. Email alerts are sent to accepted invitees and undecided invitees
      3. event is deleted
7. **Includes**
   1. Weather checks, SM
8. **Preconditions** – Event has been created with bad weather
9. **Post conditions** – Event deletes from completion
10. **Notes/Issues** - None

## Send Message (SM)

1. **SM**
2. **Objective** - The System sends a message to every attendee of the event.
3. **Priority** – High
4. **Source** – Hodor (Hodor)
5. **Actors** - User, System
6. **Flow of Events** 
   1. **Basic Flow**
      1. The System retrieves a user’s email.
      2. The System sends the email to the user.
      3. The system repeats these two steps until all of the event’s attendees have been notifified.
   2. **Alternative Flow(s)**
      1. If the event has no users, nothing will happen.
7. **Includes** - None.
8. **Preconditions** - The system has been called on to send a message concerning an event. This event must exist.
9. **Post conditions** - A message has been sent to the users event.
10. **Notes/Issues** - None.

# **Other Nonfunctional Requirements**

## **Performance Requirements**

*<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>*

***//”In this section, just say “See section 7 requirements 23-27”. And I’ll assume those //requirements are Performance related.”***

## **Safety Requirements**

*<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>*

***In this section, just say “See section 7 requirements 25-32”. And I’ll assume those requirements are Safety related.***

## **Security Requirements**

*<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>*

***In this section, just say “See section 7 requirements 35-42”. And I’ll assume those requirements are Security related.***

## **Software Quality Attributes**

*<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>*

***In this section, just say “See section 7 requirements 55-62”. And I’ll assume those requirements are Software Quality related.***

# **Other Requirements**

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

*You may not have any.*

# **System Requirements Chart**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Priority** | **Type** | **Source** | **Contained in Use(s)** | **Description** |
| EC1 |  |  |  |  |  |
| EC2 |  |  |  |  |  |
| U |  |  |  |  |  |
| U2 |  |  |  |  |  |

*< Include a* ***table*** *in this section with the following columns:*

***ID*** *– Unique requirement ID*

***Priority*** *– Priority of this requirement*

***Type*** *– Functional(F) or Non-functional(NF)*

***Source*** *– Who is most interested in this requirement (John Smith – Customer). For this project you can make it up, in reality you’ll want to capture this as you capture the requirements.*

***Contained in Use Case(s****) – Which use cases reference this requirement or which use cases when executed will perform this requirement. There may be a few functional requirements without a use-case and the non-functional requirements generally will NOT be part of a use-case (so put N/A).*

***Description*** *– The description of the requirement. “The system shall …. “*

*>*

*These requirements should match up with your use case diagrams.*

**Appendix A: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Don’t do any of these for CS421 SRS. You will create these models during the high level design deliverable.**

**Appendix B: To Be Determined List**

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*

*List here any open questions or things you know still need to be done to the SRS, but haven’t been addressed yet. (It’s okay to have things like that, especially in this CS421 project because we don’t have time to do everything.)*