CIS 422 Project 2: *Freedge Tracker*
Software Requirements Specification

Ginni Gallagher (gmg), Ellie Kobak (erk), Kalyn Koyanagi (kek),
Liza Richards (ljr), Madison Werries (mgw)

3-06-2022 - v3.0

Table of Contents

1. SRS Revision History

Date	Author	Description
1-30-2022	ljr	Created the initial document
2-14-2022	erk	Created first draft of all sections
2-16-2022	gmg	Edited document
2-16-2022	mgw	Drafted Section 3: Specific Requirements
2-16-2022	gmg	Updated Section 2 and Section 5
2-17-2022	erk	Made edits to current version
2-17-2022	ljr	Made edits to the current version, verified work done by the group
2-28-2022	ljr	Created the second version of the document
3-01-2022	gmg	Made notes of every aspect of SRS that needs to be revised
3-03-2022	erk	Created version 3 based on changes in project and feedback from initial submission
3-03-2022	gmg	Reviewed and made minor updates

3-05-2022	kek	Reviewed document and made grammatical changes
3-06-2022	erk	Final revivisions

2. The Concept of Operations (ConOps)

2.1. Current System or Situation

Freedge is the name of a non-profit organization that "[aims] to reduce food insecurity and food waste" through the construction and maintenance of public refrigerators called "freedges," which are used to "share both food and ideas at the neighborhood level," (*Freedge*). The aim of a community fridge is to reduce food insecurity and help build a stronger community. For this reason, Freedge encourages that each fridge is built, owned, and operated by one or more direct members of the community it belongs to.

In order to allow freedge caretakers to build and maintain their freedges with the best interests of their community in mind, the Freedge Organization operates under a decentralized system. The co-founder of Freedge, Ernst Bertone Oehninger, explained in an interview that under their previous model of centralization, where he had primary oversight of all freedges, they noticed widespread disengagement from each local community. For this reason, he restructured the organization to ensure freedges are owned and operated by members of the community they belong to, rather than individuals outside of it, like himself. Through this method, Oehninger notices the freedges are more successful since they can better cater to the needs of each unique community (Oehninger). However, because of the shift in organizational structure, the current Freedge communication system is composed of their old outdated model, which relies on multiple different means of communication to operate and maintain their system. Oehninger further explained that while allowing each freedge caretaker the freedom to decide how to run their community freedge is excellent for the health of the community, the decentralized structure of the organization means maintaining up-to-date information about each freedge is difficult (Oehninger). This challenge is made more complicated due to the hundreds of community freedge caretakers both in the United States and globally who have registered their freedge with the organization. Thus, Oehninger is unable to keep track of the hundreds of fridges and unable to update the website with current information.

Currently, the Freedge organization's website is out of date and not reflective of the current status of activity for each community freedge. Thus, causing difficulties for both freedge caretakers and members of a community looking to become involved or learn more about an existing fridge. Oehninger highlighted this issue in the interview, stating that he has no way of knowing which of the registered fridges are currently active, when they were last active, or if they have become temporarily inactive (Oehninger). In order to update the status of any freedge, Oehninger needs to contact the freedge caretaker directly. Manually keeping track of the vast network of freedges that have been registered with the organization is both difficult and time-consuming (Oehninger).

Additionally, in the past, Oehninger and freedge caretakers have had issues with local law enforcement and health inspectors regarding the legality of the fridges (Oehninger). For this reason, they have installed Raspberry pi sensors in most freedges to keep track of fridge traffic and the quality of food in the fridges. The sensors function in two ways. First, a sensor keeps track of every time the fridge door is open. This pleases the health inspectors because they are able to see how often someone is taking care of the fridge and ensuring no spoiled food is left in a fridge (Oehninger). Second, there is a camera sensor that displays a photo of all the contents in the fridge. This image is for community members to see what food is available in their local freedge and is supposed to be on the Freedge website (Oehninger). Freedge's sensor system is designed modularly; however, it is currently set up so only the freedge caretakers have access to the sensor data so they can send it to the health inspectors. If a Freedge organizer or a different freedge caretaker wants to view the data, they have to contact the freedge caretaker manually. The data collected by the sensors can be extremely useful when Freedge is applying for government grants. However, due to the modular system, Freedge owners have no way of accessing the data without manually checking a spreadsheet of which fridges use sensors. Additionally, the issue of not knowing which freedges are active makes determining which freedges have data on freedge activity even more difficult for Freedge organizers, such as Ernst (Denton).

2.2. Justification for a New System

There is no database to keep track of each freedge and its information. Currently, there exists a single Google spreadsheet that is filled out for each registered freedge upon initial registration of the freedge. For Freedge organizers to update their website, they must manually retrieve the data from the spreadsheet and then update the information on the website. Due to the time and effort this takes, the database of freedges on the Freedge website is often out of date. Furthermore, because all the information is stored in a spreadsheet, finding any information regarding a specific freedge requires the user to manually locate the desired fridge in the dataset, which may take significant time.

Additionally, there is no current system for the Freedge organizers to check the activity or status of registered freedges. The Freedge website has a world map of all their registered freedge locations. This map can only be edited manually, which takes too much time for Freedge organizers to maintain and therefore is not consistently reflective of the current status of every freedge. If the Freedge organizers want to update the map, they need to manually find the fridge caretaker and directly contact them regarding the status of their freedge. With the current system, Freedge organizers have no way of knowing how many freedges are actually active, nor do they consistently keep up with contacting freedge caretakers to ensure every freedge's activity is accurately reflected on their website.

Thus, storing all of the information for each freedge on one central database allows stakeholders access to activity for every freedge at any time, without needing to manually locate the contact information of the freedge caretaker in order to view data for a specific freedge.

2.3. Operational Features of the Proposed System

The notification tool and database for the Freedge Tracker will provide the leaders in the Freedge organization with easier access and maintainability for their current website and management of freedges around the world. There are three main components in the Freedge Tracker: automated tracking of freedges, accessible date for each freedge, and a notify all button for the organizer to get in touch with all freedge caretakers.

The main feature, access to notify freedges and data on when each freedge was last notified, aims to allow stakeholders and Freedge administrators to track whether or not a freedge is active. This is done through all freedges receiving an automated message asking if they are active once every 90 days. As soon as the freedge caretaker responds that the freedge is active, the fridge's counter will restart its 90 day tracker. If the caretaker does not respond, the freedge will be moved to a 'suspected inactive' list. Within the suspected inactive list, the freedge will still be notified asking their activity status, but will not move to the inactive fridge list. If the caretaker responds that the fridge is no longer active, the freedge will be removed from the active list. The database will also be updated with the active status, which will, in turn, allow Freedge Organizers to easily know the status of each freedge, and be able to update their website with up to date information.

Additionally, Freedge organizers, such as Oehninger, will have access to information for each freedge. Instead of using a Google Spreadsheet, the data for each freedge will be stored in a database. The database has information regarding the address, caretaker name and contact information, date of installation, freedge identification number, and the active status for each freedge. Not only does a database create a more organized system for Oehninger and other Freedge organizers, but a database also allows for modularity to incorporate potential future functionality in addition to our proposed notification system.

2.4. User Classes

There are two main user classes for the Freedge Tracker System.

1. **Administrators**
   The main user class for this system is for internal Freedge personnel referred to as administrators. The administrator will have access to an application that is automatically maintained from a database containing all the information of fridge use that is currently stored in a spreadsheet.

2. **Caretakers**
   A secondary user class consists of fridge caretaker notifications. These notifications, sent by the main system or by the instruction of a Freedge administrator, require explicit consent to be received from fridge caretakers and their input to respond. For this prototype, the caretaker notifications are sent to the main window of the user running the system.

2.5. Modes of Operation

The mode of operation for the Freedge Tracker is a prototype for Freedge stakeholders and administrators to make maintenance easier and increase organization for the group. The Freedge Tracker also helps Freedge administrators have tangible information on how many freedges they have active, which will help them with both organization and for applying to federal grants for their organization.

The fridge caretaker mode of operation consists of notification communications between the Freedge Tracker system and the fridge caretakers. This communication takes place between the system and the caretaker's personal device, being a computer or a smartphone via email or SMS. The prototype implemented does not have the ability to send SMS and emails due to lack of funding, but it does have popup notifications visible in the administrator interface.

2.6. Operational Scenarios (Also Known as "Use Cases")

Use Case: The Freedge administrator checks the status of a particular fridge.

Brief Description: The Freedge Tracker system has automatically noted that the status of a particular fridge has not been updated in the past 90 days. The system sends a popup with a notification and in the final version, a text or email to check the status of the fridge, updating the system according to the freedge caretaker's reply.
Actors: A freedge caretaker.
Preconditions: The status of a particular freedge has not been verified in 90 days. That freedge caretaker has previously agreed to receive notifications regarding the status of their freedge.

Postconditions: The status of the freedge is contingent on the freedge caretaker's response to the system's notification.

Visual Representation of Concept:

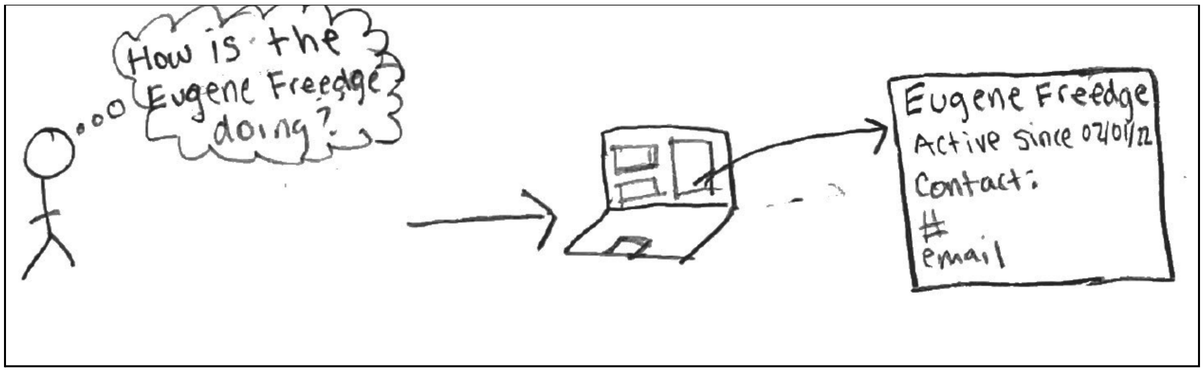

Figure 2.6.1
Example of how the organizer will be able to access the status of a fridge.

Use Case: The Freedge administrator views freedge data or contacts a specific freedge.

Brief Description: A Freedge administrator seeks additional data regarding the usage of a particular freedge. Upon viewing the data compiled by the system, the Freedge administrator may decide to contact the freedge caretaker further and inquire whether the freedge is still active.

Actors: Freedge administrator.

Preconditions: The Freedge administrator is curious about the current status and activity level of a freedge.

Steps to Complete the Task:

1. The user selects which freedge they want to view from the main menu drop-down on the user interface.
2. The user can either close out if they are done looking at the data, or press notify to notify the freedge, asking if it's currently active.

Postconditions: The user has viewed the data or contact information for a specific freedge and notified the freedge caretaker if they wanted to know whether or not the freedge is active.

Visual Representation of Concept:

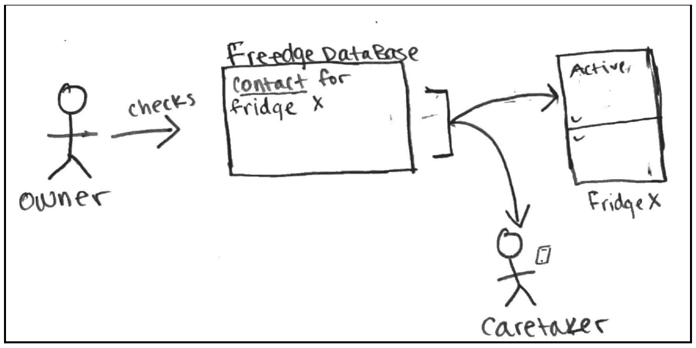

Figure 2.6.2
Example of how the organizer will be able to access the status of any fridge.

Use Case: The Freedge administrator wants to notify all freedge locations

Brief Description: If any Freedge administrator needs to notify all freedge locations, they can send a predetermined automated notification inquiring about the status of the caretaker's freedge to all registered freedges in the system.

Actors: Freedge administrator.

Preconditions: The Freedge administrator wants to check the status of all currently active freedges. The message sent out is automated and predetermined.

Steps to Complete the Task:

1. The Freedge administrator selects the "Notify All" button from the main menu.
2. A confirmation will appear, prompting the administrator to confirm they want to notify all freedges.
1. The system will display "All freedges have been notified" once notifications have been sent.

Postconditions: The Freedge administrator has sent each freedge caretaker a notification.

Visual Representation of Concept:

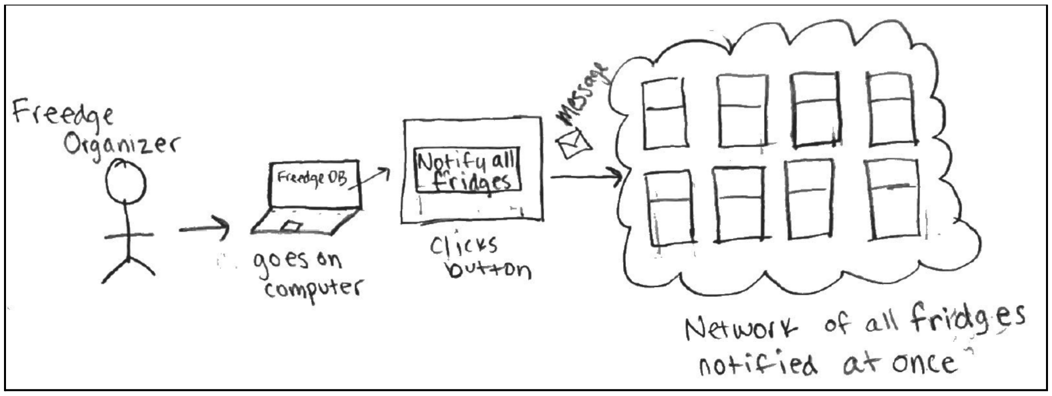

Figure 2.6.3
Example of how the organizer will be able to notify all fridges at one time.

Use Case: The Freedge administrator views recent notifications.

Brief Description: If a Freedge administrator is curious about when one or more freedges were last notified of their activity status and what the response was, they can view a table containing this information.

Actors: Freedge administrator

Preconditions: The Freedge administrator wants to review when each freedge was last notified.

Steps to Complete the Task:

1. The Freedge administrator selects the "View Last Notification" button from the main menu.
2. The Freedge administrator will see the table of all freedges, the timestamp of their last notification, and the status of the previous notification reply.

Postconditions: The Freedge administrator has viewed the latest notification status for all freedges.

Visual Representation of Concept:

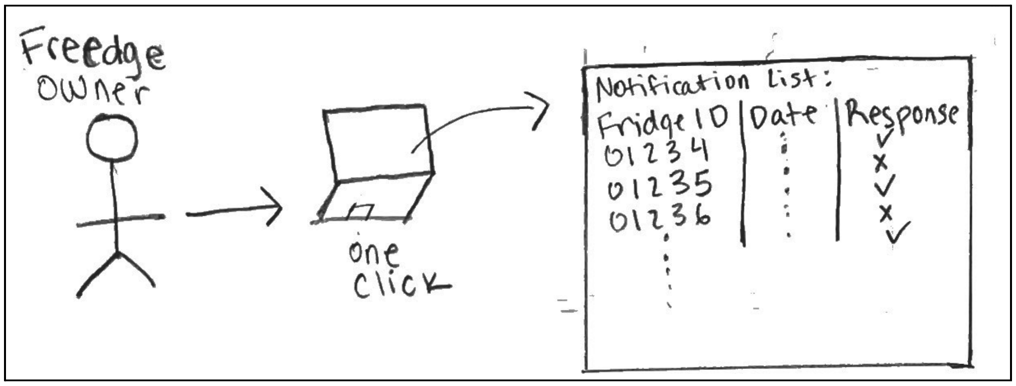

Figure 2.6.4
Example of how the organizer will be able to view past notifications.

3. Specific Requirements

The specific software requirements for the Freedge Tracker system are described here, where Section 3.1 describes the functional requirements of the software, and Section 3.2 describes the non-functional software requirements.

The functional and non-functional requirements have been further divided into three distinct categories based on their hierarchical importance:

1. ***Absolutely required:*** These are the core software requirements which must be met in order for the Freedge Tracker system to function as expected.
2. ***Not absolutely required:*** These are the requirements that are essential but not necessarily vital for Freedge Tracker to function.

3. ***Future considerations:*** These are the software requirements which have not yet been included as part of the Freedge Tracker but could be considered in future implementations.

3.1 Functional Requirements

[FR] ***Absolutely required:***

1. The system will read the initial database from a comma-separated CSV file.
2. The system will update the database to reflect user responses.
3. The system will automatically create a new database using information read from an input file. When the number of days between the current date and the date of the last time the fridge's status was updated, the system will automatically update the status of the freedge.
4. The system will send automated notifications to freedge caretakers based on the date since the last status update of that freedge.[1]
    - 5.1 The system will only send notifications to those caretakers who have given explicit permission to receive such notifications.
    - 5.2 The default time between notifications will be 90 days.
5. The system will be able to receive responses from caretakers.
    - 6.1 After sending out a message to verify if a freedge is still active, the system should be able to receive and process “YES” or “NO” responses from freedge caretakers.
6. The system will automatically update the freedge status from active to inactive in the Freedge database if no response from a freedge caretaker is received after the final warning notification.
7. The system must maintain an up-to-date counter of the number of days between notification timestamps for each registered freedge.
8. The system will keep track of the timestamp of the most recent notification to a freedge caretaker and any message replies.

[FR] ***Future considerations:***

1. Alternative notification format.
    - 1.1 The system will be able to send out automated email messages to fridge caretakers in addition to SMS messages.
2. The system can host a Slack channel for notifications and updates on when a freedge becomes inactive so that Freedge administrators and freedge caretakers can receive up-to-date information on inactive freedges.
3. The system can accept freedge administrators’ requests to manually deactivate a freedge and remove it from the list of active freedges.

[1] In the prototype, notifications will be popups with instructions on how to incorporate emails and SMS for the future working system.

4. Freedge data collection and display
    - 4.1 The graphs will display the usage data collected from Raspberry Pi sensors, which are included by some of the freedge caretakers.
    - 4.2 When an administrator selects a specific freedge from a list of those which have data-collection set up, usage graphs will automatically generate from the sensor data input file for the Freedge administrators to view.

    Valid ranges of inputs and outputs: The inputs of the graph must contain the location and time from data from the sensor data. The graphs will output simple line graphs created using Pandas.

    Units of measure: If a fridge is inactive, no graph will be created. If a fridge is active, the graph will be generated using whole number inputs for the amount of usage and date.

    Data formats: String of text paired with a timestamp.
5. Saving Freedge Raspberry Pi Data Graphs
    - 5.1 The Freedge administrator will be able to save the usage summary graphs into either a data summary txt file or as a PNG image.

3.2 Non-Functional Requirements

[NFR] ***Absolutely required:***

1. The Community Fridge Database will be implemented using PySQLite, which, as described in the Python Package Index, is an interface to the SQLite embedded relational database engine, and has been part of the Python Standard Library since Python version 2.5.
2. All of the information about the fridges registered in the Freedge system is to be stored in a CSV file.
    - 2.1 The spreadsheet contains the location of the fridge, email and phone number of the fridge caretaker, and the date the fridge was registered. The data from the file will be parsed to use for sending notifications and locating the fridge on the map.
    - 2.2 Valid ranges of inputs and outputs: The inputs for the file data will be strings parsed from the CSV or txt file. All inputs should be valid, meaning they have been verified as the correct phone number by a Freedge organizer upon inputting the information into the database.
    - 2.3 The installed dates will be formatted in the format DD-MM-YYYY
3. At most, 20 user actions will be required to compile the code and run the program.
4. An experienced computer programmer should not require more than 30 minutes working alone with the submitted materials to compile and run the code.
5. The system must run on Macintosh OSX 10.13 (High Sierra) or 10.14 (Mojave).

6. All system-related and system-development-related documents that are intended for human reading must be saved in either plain text or PDF.

[NFR] ***Not absolutely required:***

1. The system can accept and differentiate between multiple input file types.

[NFR] ***Future considerations:***

1. Integrating Google Maps with our system so that accessing an up-to-date map of worldwide freedges is more accessible.

3.3 External Interface Requirements (Inputs and Outputs)

Must-Have:

Sent Notifications (output)

Description: Predetermined strings of text that are both automatically and manually sent out to fridge caretakers.

Source of output: Notifications are triggered by a button push or automatically sent out based on timestamps.

Valid ranges of inputs and outputs: The input must be from a timestamp in numerical format. The output is a string from a defined text constant.

Units of measure: The measurements will be boolean values for if the notification has been sent or not.

Data formats: String of text paired with a timestamp.

4. References

Denton, Sabrina. Interview. By Liza Richards and teammates. 24 February 2022.

Freedge, https://freedge.org/.

IEEE Std 1362-1998 (R2007). (2007). IEEE Guide for Information Technology–System Definition–Concept of Operations (ConOps) Document. https://ieeexplore.ieee.org/document/761853

IEEE Std 830-1998. (2007). IEEE Recommended Practice for Software Requirements Specifications. https://ieeexplore.ieee.org/document/720574

ISO/IEC/IEEE Intl Std 29148:2011. (2011). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/6146379

ISO/IEC/IEEE Intl Std 29148:2018. (2018). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/8559686

Faulk, Stuart. (2013). *Understanding Software Requirements*. https://projects.cecs.pdx.edu/attachments/download/904/Faulk_SoftwareRequirements_v4.pdf

Oehninger, Ernst Bertone. Interview. By Madison Werries and teammates. 14 February 2022.

Oracle. (2007). White Paper on “Getting Started With Use Case Modeling”. Available at: https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf

van Vliet, Hans. (2008). *Software Engineering: Principles and Practice*, 3rd edition, John Wiley & Sons.

5. Acknowledgements

This template was given to the UO CIS 422 class by Anthony Hornof. This template is similar to a document produced by Stuart Faulk in 2017, and uses publications cited within the document, such as IEEE Std 1362-1998 and ISO/IEC/IEEE Intl Std 29148:2018.