Social Distance Assistant

Software Requirements Specification

Group 5 - 6-1-2020 - v2.01

Irfan Filipovic (IF), Shane Folden (SF), Man Him Fung (MF), Mason Jones (MJ), Siqi Wang (SW)

Table of Contents 1

1. SRS Revision History

Date	Author	Description
6-1-2020	Group	v2.01 - Finalized all revisions
5-29-2020	IF	v2.0 - Updated sections according to feedback
5-15-2020	Group	v1.3 - Added all additional information and revised entire document
5-14-2020	Group	v1.2 - Transferred and revised all applicable information from Project 1
5-11-2020	Group	v1.1 - Created simple Project 2 SRS skeleton

2. The Concept of Operations (ConOps)

This application is designed to be used by people that are going to restaurants and stores during a pandemic in order to help them figure out what time of day is optimal to minimize human interaction and time spent waiting in line. The application does this by automatically collecting each user's location data once given authorization. It also provides an option to submit data on how long they waited in line if they were at a business. This data is stored in a database and then automatically displayed on a map located on a publically accessible website. To help protect privacy, a user is able to indicate when they are at home to offset their location data.

2.1. Current System or Situation

A new coronavirus; Covid-19, was first noticed spreading in November 2019. By March of 2020, everyday life in the United States, along with the rest of the world, had been dramatically changed. Stay-at-home and social distancing orders led to a significant decrease in social interaction in an effort to minimize human contact. Some scientists estimate that one person infected with the new coronavirus will infect an average of 3.3 other people, so eliminating unnecessary interactions is of extreme importance to limiting the spread of the virus (Khan). Despite these orders many individuals need to access essential services like grocery stores, hardware stores, and restaurants.

MD2K, the Mobile Sensor Data-to-Knowledge Center of Excellence, is a conglomeration of educational institutions that, “aims to lay the scientific foundations for turning the wealth of mobile sensor data available through new and rapidly evolving wearable sensors into reliable and actionable health information, and contribute to the vision of predictive, preventive,

personalized, participatory, and precision (P5) medicine" (MD2K). MD2K released a mobile application named *mContain* to assist users in maintaining social-distancing; *mContain*, released on April 10, 2020, notifies a user of their possible past encounters with confirmed carriers of Covid-19. *mContain* uses Bluetooth to detect proximity with other users of the application. After receiving a notification of a possible exposure to Covid-19 the user can then decide to remain at home or consider taking a test.

2.2. Justification for a New System

Our aim is to preempt the spread of Covid-19 by allowing users to view information pertaining to the spread of individuals which can be used in addition to other systems. Providing information which may help users maintain their health is the main concern. Currently there are some other applications that assist users regarding Covid-19, however they do not address our concerns. To address these concerns a user may use our application in addition to others.

mContain, developed by MD2K, only tracks encounters with users that are in a 6 foot proximity. While this does allow users to quickly obtain knowledge of their possible exposure to Covid-19, it does not assist in the prevention of exposure. To develop a system that assists in slowing the spread of Covid-19, we aim to provide useful information prior to exposure rather than after. This information, such as average waiting time at a certain store or the number of individuals at a location, would allow a user to make educated decisions on how to best approach social-distancing while continuing daily actions.

> "But the country's virus reproduction rate - known as "R" - which measures how many people the average person with Covid-19 infects has bounced back to just below one.
> That means one person with the virus infects one other on average and earlier this month, the rate was at 0.7." (The Sun)

As shown by this article, easing social distancing restrictions and reopening popular businesses; both of which promote people to come in close proximity of each other, has shown to promote the spread of the virus (Tahir, Tariq). Shopping at non-peak times, buying in bulk, reducing fast-food intake, and reducing/eliminating social contact are all actions people are able to take in order to minimize their potential exposure to Covid-19 in addition to following proper sanitary procedures. Our application will assist in simplifying the user's guessing in relation to the best time to visit a store or restaurant.

2.3. Operational Features of the Proposed System

The system will assist users in anticipating wait-times and pedestrian density when visiting popular destinations. We will be tracking user data on a mobile application and

uploading this data to a database to use in visually representing foot-traffic information at popular destinations, addressing that *mContain* does not provide information that assists in preventing the contraction of Covid-19. This functionality allows users to utilize our app alongside *mContain* in order to get as much information as possible pertaining to Covid-19. The user will have access to a map on a webpage which will denote other user's locations, average hourly foot traffic at popular destinations, average time spent by users at a location, average time spent in line at a location, and the number of users at said location. The existing version of our mobile application, generated from CIS 422's first project, will be updated to address user experience issues. An issue that arose indicated a user was not aware of application status; we aim to improve user experience by making the application's status apparent by visually indicating its current status on a phone's screen. These statuses may be along the lines of "connected to database," or "tracking and obfuscating location." To assist in testing our application, our development team will have access to an in-app interface that will allow for the changing of frequency at which packets containing data are sent as well as allow for the creation of fake data-containing packets.

Website Features:

- Automatic markers for each user are displayed on the map
    - User markers display: the phone id, the time the data was sent, the longitude/latitude, and the time spent at the location
- A custom Mapbox layer highlights popular stores and restaurants
- Per location (store or restaurant), the average time spent at a location will be displayed
- Per location, the number of users at a location will be displayed
- Per location, the average foot traffic per hour, organized by day of the week, will be displayed
- Per location, the average time spent in line at a location will be displayed

iPhone Application Features:

- Status indicator showing that location data is being tracked
- A packet is sent to the system upon enabling location tracking
- User page to submit line time information
    - Allows a user to enter the address of a business they visited as well as enter the amount of time they waited in line there
- Protected back-end page allowing authorized users to create test packets and choose the frequency at which packets are sent

2.4. User Classes

- Development Team
    - Creates, updates, and maintains the application
- Everyday Users
    - Download the application
    - Share location data
    - View the highlighted popular locations on the map
    - View visual anonymous data on the map
    - View average peak foot traffic data per location
    - View average time people spend at a location
    - Enter their time spent in line at a location
- System Administrator
    - Granted protected access with a key from developers
    - Test the application, using the secure interface created.
    - Create and send test packets
    - Adjust frequency of regular data sending
    - May use the application just like Everyday Users.

2.5. Modes of Operation

- Testing Mode
    - Available on the app via password-protected page provided by the developers
    - Provides operations to manage time location intervals
    - Provides a way to send test data packets and view database response
- General Mobile App Use (Interactive)
    - Available on the app to Development team and Everyday Users
    - Provides the ability to track location data and upload it to the database automatically
    - Provides the ability to toggle location tracking and home location obfuscation
    - Provides a way to input and upload the amount of time spent waiting in line at a location
- Passive Mobile App Use (Running in background)
    - Available on the app to Development team and Everyday Users
    - Has the ability to track location data and upload it to the database without user interaction

- Active Viewing of Visualized Data Use
    - Available on the website to the Development Team and Everyday Users
    - Provides the ability to view the foot traffic statistics of locations on a map

2.6. Operational Scenarios (Also Known as “Use Cases”)

Use Case: Passively tracking location data

Brief description: This use case describes how a user’s location data will be tracked and stored.

Actors: Any user

Preconditions:

1. The user must have Xcode 11 to install the app which requires a version of macOS 10.14.4 or later
2. The user has an iPhone 5 or later with iOS 12.0 or later
3. The user has access to the internet
4. The user has location services enabled on their mobile device
5. The user is successfully connected to a database

Steps to Complete the Task:

1. The user downloads the application from XCode and opens it
2. The user taps 'Allow While Using the App' when prompted about location use
3. The user toggles the 'Start Location Tracking' button which starts tracking and uploading the users location
4. The user toggles the "At Home" button when they are at home, toggling it again when they leave their home
5. The user leaves their device powered on and with them
6. The user toggles the 'Start Location Tracking' button into the off position when they are done collecting data

Postconditions:

The user has allowed the application to run uninterrupted, has maintained a powered mobile device, and has their device with them at all times. Data has been tracked.

Use Case: Entering line times at businesses

Brief description:

This use case describes how a user will be able to enter the amount of time they spent waiting in line at a store, restaurant or other business.

Actors: Any user.

Preconditions:

1. The user has Xcode 11 to install the app which requires a version of macOS 10.14.4 or later

2. The user has an iPhone 5 or later that is running iOS 12.0 or later
3. The user has access to the internet
4. The user has location services enabled on their mobile device
5. The user has the app open to the home screen
6. The user is successfully connected to a database
7. The user has authorized the app to access their location

Steps to Complete the Task:

1. The user presses the 'Enter Time Spent in Line' button
2. The user enters the address of the business they visited
3. The user enters the amount of time they spent waiting in line at the business (in minutes)
4. The user presses the 'Submit' button to submit the business they are at along with their time spent in line
5. The user presses the 'Back' button to return to the home page

Postconditions:

The user has visited a store, entered the name of the store and the duration of time they had to wait there, and uploaded that data to our database.

Note:

Currently users must enter the information while at a store, a functionality which may be automated with an update following the evaluation of how the data is improving user experience.

Use Case: View raw location data in the database

Brief description: This use case describes how a developer views information in MySQLWorkbench.

Actors: Development team

Preconditions:

1. The member installed and configured the mysql/mysqlctl packages
2. The member is running the mysqlctl and mysql processes
3. The member has created and configured a mysql database
4. The member has created a user account for themselves using the mysql process
5. The member has installed MySQLWorkbench

Steps to Complete the Task:

1. The member opens MySQLWorkbench
2. The member selects “Database” → “Connect to Database” and enters their database’s hostname and port, their username and password, then selects the “OK” to use this information
3. The member runs their desired queries

Postconditions:

The member has started the mysql process, established a connection to their database using MySQLWorkbench, and is able to view/modify data.

Use Case: View location data on the webpage

Brief description: This use case describes how a user views location data information

Actors: Any user

Preconditions:

1. The user is on a computer
2. The user has installed the http-server package
3. The user has run the http-server package from inside of the folder containing the application files
3. The user has configured all supplied files correctly
3. The user has an internet browser downloaded on their device

Steps to Complete the Task:

1. The user opens their internet browser
2. The user navigates to the address specified by the output of running http-server
3. The user navigates to the proper file

Postconditions:

The user has the internet window open and is able to see the map presented on their screen.

Use Case: Dev mode

Brief description: This use case allows developers access to the developer mode screen where they are able to change the frequency of sending packets and create fake location data packets.

Actors: Development team

Preconditions:

1. The developer has the app downloaded and open to its home screen
2. The developer has access to the development password

Steps to Complete the Task:

1. The developer presses the dev mode button
2. The developer inputs the development password
3. If the password is correct, the dev mode screen will open, if not the user will not be permitted into dev mode

Postconditions:

The developer is able to change the frequency at which packets are sent and create fake location data packets.

3. Specific Requirements

MUST HAVE:

1. Display user app state
    1.1. Update user interface to reflect application status
        1.1.1. The application must notify the user when the database is connected, when location is being tracked, and if location is being obfuscated. This allows users to be aware of the actions taking place on their device.
        1.1.2. On a failed connection to the database or if location tracking is enabled but not operating, the user will be notified of a failed state so they know to request assistance.

2. Map hotspots and markers
    2.1. Update the Mapbox map to display the proposed features, see “2.3. Operational Features of the Proposed System” for the full list of website features.

3. Testing interface on the application
    3.1. Allow authorized developers access to an in-app page for testing functionality of the system, such as creating a data packet with current location and sending data on intervals.
        3.1.1. Authorized developers are allowed to create test packets and alter the frequency at which packets are sent. The user can enter the desired latitude and longitude. After sending data a response from the php script will be interpreted and displayed to the user as an alert.

SHOULD HAVE:

4. Manage dependencies
    4.1. Update mobile application to support iOS 12 in addition to iOS 13 so more users are able to use the app.

3.1. External Interfaces (Inputs and Outputs)

MUST HAVE:

User Data Collection

1. User Data Collection
2. This is the data that will be stored and used for our application. All system functionality is based on this data

3. **Input -** Location data from user cell phones | **Output -** Location data is formatted and stored in the database
4. User I.D.\tDate\tTime\tLatitude\tLongitude\tTime at Location\n
5. Units of measure:
    - Latitude and Longitude: degrees
    - Time: hours and minutes
    - Time at Location: minutes
6. Tab-delimited text

Visual Representation of Data

1. Visual Representation of Data
2. This enables the user to view the data easily in a familiar way
3. **Input -** .json file **| Output -** Markers in the form of circles positioned at specific longitudes/latitudes on a map hosted by Mapbox
4. A valid .json or .geojson file
5. Units of measure: N/A
6. Website (.html) with a map which connects to the database using a .php file

Backup System

1. Backup System
2. Creates and stores a copy of all location data. This is to ensure there is a minimal loss of data if the primary database goes down
3. **Input** - mySQL database file | **Output** - mySQL database file
4. mySQL database file
5. Units of measure: N/A
6. mySQL database → .sql file

Mobile Developer Testing Interface

1. Mobile Testing Interface
2. Allows developers to test sending data from an iPhone by offering the abilities to change the frequency that location data packets are sent as well and create location data packets using made up data.
3. Location Data Packets:
    - **Input** - Numerical frequency of sending location data packets and fake location packet data | **Output** - A fake location packet will be sent and/or the frequency at which packets are sent will be adjusted

4. Current location

5. Units of measure: Packet frequency: seconds
6. Creating fake packets: User I.D.\tDate\tTime\tLatitude\tLongitude\tTime at Location\n

SHOULD HAVE:

Time in Line at Location Interface

1. Time in Line at Location Interface
2. This page allows users to enter the location they are currently at as well as the time they spent in line before leaving the location
3. **Input** - Location name, address, time spent in line | **Output** - Data is POSTed from the mobile application to our .php page which then sends it to the database
4. Time spent in line: integer, Location name and address: text string
5. Units of measure:
    - Time spent in line: seconds
6. Time spent in line: 00:00:00, Location name: string, Location address: string

COULD HAVE:

User Login

1. User Login
2. This is how a user will receive an assigned user-id and verify their access to the id
3. **Input -** Username and password **| Output -** Token verifying access
4. User inputted text string
5. Units of measure: N/A
6. Token string

Mobile Visual Representation

1. Mobile Visual Representation of Data
2. This enables a user to view the website's map through the mobile application
3. **Input -** Link to the website's map display **| Output -** Viewable map
4. String identifier for visual representation web page
5. Units of measure: N/A
6. A view on the app that displays our website's map

3.2. Functions

- **User Data Collection**
    - Validity checks on the inputs:

        - Location services accepted, and data provided by device
    - Sequence of operations in processing inputs:
        - Obtain current time
        - If an interval of 5 minutes has passed, obtain necessary data
        - Insert data to a request body
    - Responses to abnormal situations, including error handling and recovery
        - The location will not be collected if services are not authorized, they will prompt again on opening
        - Application will not shift orientation if the device rotates
        - User input will be allowed once for each option, before being enabled again
    - Relationship of outputs to inputs, including:
        - input/output sequence
            - Authorize location services
            - Enable location tracking, pinged every 5 minutes
            - Data sent in a request to the database webpage
        - formulas for input to output conversion
            - Convert coordinates to four decimal places by utilizing rounding and shifting
            - Obtain formatted dates and times from current day and time via Swift functions

- **Line Time Collection**
    - Validity checks on the inputs
        - Ensure line time is only entered for businesses in a close proximity to the user
        - Ensure line time is a non-negative integer
    - Sequence of operations in processing inputs
        - Makes call to Apple's MapKit API to retrieve address of a user selected business
        - If the Enter button is clicked, line time data is inserted into a request body
    - Responses to abnormal situations, including error handling and recovery
        - The app will reject multiple requests to the same business if they are submitted in under 10 minutes to prevent spam
        - The app will confirm a database is connected before allowing user to enter the line time collection screen
        - The app will limit a user to 3 entries per minute to prevent spamming the form
    - Relationship of outputs to inputs, including:

        - Input/output sequences
            - User types in business name
            - User selects the correct address of the business as prompted by MapKit
            - User enters time spent in line (minutes)
            - Data is sent in a request to the database webpage
        - Formulas for input to output conversion
            - N/A

- **Change Database in Dev Mode**
    - Validity checks on the inputs
        - Ensure database name and port are entered in the text fields
        - Ensure port is an integer between 0 and 65535 (inclusive)
    - Sequence of operations in processing inputs
        - Attempt connection to MySQL database with entered name
    - Responses to abnormal situations, including error handling and recovery
        - Program will throw an error if it is unable to successfully connect to the database and prompt the user to retry or exit
    - Relationship of outputs to inputs, including:
        - Input/output sequences
            - Enter and submit a database name and port in the dev interface.
            - Display successful connection to the user unless an error is raised as defined above.
        - Formulas for input to output conversion
            - N/A

- **Backup System**
    - Validity checks on the inputs
        - Mysql server connection is accepted or dropped
    - Sequence of operations in processing inputs
        - Auto-fetch data every 6 hours (shell script) → store in local host→ push to GitHub every 6 hours (shell script)
    - Responses to abnormal situations, including error handling and recovery
        - Backup data and the data pushed to GitHub may not be the same time. After pushing the file it will show up late on GitHub. It will push the files again after the gaining data from mysql database
    - Relationship of outputs to inputs, including:
        - input/output sequences
            - Backup to localhost

        - Input: Mysql host, port, username, password
        - Export data into .sql file by script every 6 hours
        - Output: .sql file that is stored in localhost
      - Upload to GitHub
        - Input: .sql file
        - Push .sql file to GitHub by script every 6 hours
        - Output: .sql file that is stored on GitHub
    - formulas for input to output conversion
      - Input: Mysql connection information -> bash script -> store sql file in local host -> bash script -> output: push to to github
- Database setup
  - Set up mysql database on ix-dev
    - CREATE DATABASE CIS422_Project1
      - User name: manhimf
      - Host: ix-dev.cs.uoregon.edu
      - Port: 3114
- Crontab
  - Used to schedule and run the script automatically to backup database
    - Backup data every 6 hours and push to github
- Backup github database
  - It is where to store the backup up sql file https://github.com/manhimf/CIS422_Project1_backup_v2.git

- **Visual Representation of Data**
  - Validity checks on the inputs
    - Proper format directly from database
  - Sequence of operations in processing inputs
    - Get data file → Send contents to a new HTML div → Data is processed into a forEach loop → create new HTML divs for each data → create Markers on our map
  - Responses to abnormal situations, including error handling and recovery
    - The result of our query is formatted using JSON.parse() to ensure it will be usable
  - Relationship of outputs to inputs, including
    - input/output sequences
      - Input: Input .json file, Output: json formatted text in a div
      - Input: json formatted text in a div, Output: HTML divs and mapboxgl.Marker objects
    - formulas for input to output conversion

- Input: SQL query → while($row = $result2->fetch_assoc()){fwrite()...} where we insert data points into a manually created json format
- Output: point, average time showed on map

3.3. Usability Requirements

The system will be designed to default require low interaction from the user. While operating in the background, the system will need to record the user's geo-location data to the database in order to be effective. During the interval between recording data, the system will halt most of its actions to reduce resource utilization.

To maintain privacy while at home, a button exists to toggle whether current location data is currently obfuscated. It is only to be used while a user is at home, and a consistent random value between 0.05 and 0.25 miles is applied to latitude/longitude when active.

Labeling and organizing the buttons from which location tracking as well as obfuscating location are started will increase the user's experience interacting with the application. Once these actions are toggled to active, the application will display if the toggle was successful and notify the user of the application's actions.

Data will be presented in a format that can be interpreted by a reader, following the tab delimited format supplied in the project description. There will also be a web page available to see current user locations along with other relevant data. The webpage supplying current user locations will also present information relating to popular locations like grocery stores. If a user visits a store they have the option to enter the time they waited in line, which will then be displayed on the map.

These requirements ensure user interaction will be minimal while still recording, displaying status of, and maintaining location data; yielding the desired outcome. Furthermore, this helps in preventing a negative user experience by not significantly increasing the load placed on their mobile device.

3.4. Performance Requirements

- 100% of users connected to the geospatial database with location tracking enabled will have their location displayed on the website

- Location data will be collected and posted every 5 minutes unless otherwise specified in the Dev mode settings
- The system will perform a scheduled backup of the database every 6 hours
- Mobile CPU utilization will remain below 5% to ensure minimal loss of battery charge for the user
- The outputted tab-delimited text file will contain 100% of recorded data

3.5. Software System Attributes

Reliability: Location data must be accurate to ensure the user can depend upon the information provided. A user's location will be tracked using the device's location services provided and maintained by Apple. The longitude and latitude will be shortened to less decimal places and later obfuscated, but other data will not be altered.

Usability: Having an easy to interpret map displaying foot traffic data and the ability to view additional information about popular locations will improve the user's experience. We will maintain an easy to use interface by conducting direct observation tests on beta users.

Maintainability: Hosting a website and real time location database requires minimal downtime to ensure users have access to the system. The system is designed to maintain and observe direct connections to modules. Upon an error in the system, it will be indicated which connection or module is issuing the error.

Testability: To ensure the connections between modules are successful and data is being sent a testing interface will be provided. The interface will allow for changing the frequency that data is sent to the database as well as changing the database to which data is sent. This will allow for the operations to be observed, tested, and maintained. The system's webpage will display live updates and can be simply tested by opening the site.

4. References

Tahir, Tariq. "Germany Could Have to Bring BACK Lockdown as Cases Surge Days after Reopening." *The Sun*, The Sun, 30 Apr. 2020, https://www.thesun.co.uk/news/11505630/germany-bring-back-coronavirus-lockdowns-cases-surge/.

The MD2K Center of Excellence at The University of Memphis. “mContain.” *mContain*, https://mcontain.md2k.org/.

Khan, Amina. "Coronavirus tips: The do’s and don’ts of social distancing" https://www.latimes.com/science/story/2020-03-18/coronavirus-tips-the-dos-and-donts-of-social-distancing

https://classes.cs.uoregon.edu/20S/cis422/P1/LATimes.pdf

5. Acknowledgements

https://app.diagrams.net/

This web page was used to create the diagrams in the document.