Project 1
Software Requirements Specification

Group 5 - 4-27-2020 - v2.0

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

Table of Contents

1. SRS Revision History

Date	Author	Description
4-10-2020	group	Gutted default text, revised document to be the skeleton for Sunday's meeting.
4-12-2020	group	Expanded on SRS skeleton. Established color code, fully marked up SRS v1.
4-15-2020	mj	Cleaned up for presentation.
4-23-2020	mj	More clean up.
4-25-2020	IF	Start cleanup according to feedback.
4-27-2020	MF, IF	filling up function, fixing all sections
4-27-2020	Group	Fix sections.

2. The Concept of Operations (ConOps)

The goal of this application is to collect a user's location data, store it in a database, and display a collection of all users' data in a visually meaningful way. This database will produce tab-delimited text files of the location data. The database will be backed up periodically to ensure minimal data loss.

2.1. Current System or Situation

A new coronavirus, Covid-19, was first noticed to be spreading in November of 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 efforts to minimize human contact. Some scientists estimate that one person infected with the new coronavirus will infect another 3.3 people. Despite these orders many individuals need to access essential services such as grocery stores, which results in long lines and conglomerations of people.

Currently, the best application to assist social distancing measures is Trailcheck. Trailcheck is very similar to our application, but is limited to hiking trails and requires users to manually enter their location data. Google Maps also functions similarly to our application, but the only relevant reports deal with car-related road traffic and peak foot-traffic in stores. The Google Maps application only functions for users that utilize google maps as an application for their navigation.

2.2. Justification for a New System

Trailcheck: This application does not have a large user base, likely because it requires users to actively enter their location data. It focuses solely on hiking trails whereas our application will deal with many popular destinations. These destinations will include grocery stores, pharmacies, hiking trails, public parks, and more.

Google Maps: While it has many useful functions, it's only *situationally relevant* functions are displaying car-traffic by road and peak average foot traffic in stores by time. It does not consider the maximum allowed number of people in a store, line lengths, or average idle/wait times. Traffic is never considered for non-roads, such as hiking trails or popular stretches of sidewalk (such as one that stretches through the University of Oregon campus). If an individual were to use Apple maps or any other service they would not benefit from the time displays.

To provide data for the general public about the general public it is important that there exists a system designed to record data regarding social distancing so as to provide features not relevant or provided in other cases. Considering issues with the available systems and developing further requirements will produce a system specific to maintaining health by adhering to orders of minimal social contact.

2.3. Operational Features of the Proposed System

First, our team will track and store location data with the plan of assisting communities in keeping peak foot traffic low and human interaction to a minimum. The user will be able to designate when they are at home and have their location shifted by a designated amount. A database for the location data may be designed with the function of storing raw data and outputting formatted text files. A website may be created in order to visualize the data on a map, and a system will be implemented to backup our database.

While this won't directly fulfill user needs, this will lay the framework for *Project 2*. *Project 2* will greatly enhance the practicality and usability of the location data.

2.4. User Classes

- *Development Team*
    - Creates, updates, and maintains the application
- *Everyday Users*
    - Download the app
    - Open the app and press the start data collection button

- Click allow location tracking when prompted.
- Designate when they are at a home location.
- View visual anonymous data at website

2.5. Modes of Operation

- Developer Mode (Unconfirmed)
  - Available to Development team
  - Provide operations to manage the database and collection services
  - Provide ability to change function, representation, or format of data
  - View raw data in the database
- General Use (Interactive)
  - Available to Development team, and Everyday Users
  - Track location data, and upload to database
  - Toggle location tracking and home location obfuscation
  - View visual anonymous traffic data
- Passive Use (Running in background)
  - Available to Development team, and Everyday Users
  - Track location data, and upload to database

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

Use Case: Passively storing 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 has access to the internet.
2. The user has location services enabled on their mobile device.
3. The user has an iPhone 7 or later with iOS 13 or later.

Steps to Complete the Task:

1. The user downloads the application and opens it.
2. The user taps 'Allow While Using the App' when prompted about location use.
3. The user taps the 'Start button' which starts location tracking.
4. The user taps the "At Home" button when they are at home and the 'not at home' button when they are not.
5. The user leaves their device powered on and with them.
6. The user presses the "stop button" 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: View raw location data in the database

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

Actors: Development Team Member

Preconditions:

1. The member has access to the internet and is able to connect to ix-dev.
2. The member is on a computer, not a phone, tablet, or another device.
3. The member has been assigned a username and password on the utilized ix-dev account.
4. The MySQL server is running.

Steps to Complete the Task:

1. OPTIONAL: The member opens MySQLWorkbench.
2. The member connects to the database via ix-dev.
3. The member runs their desired queries.

Postconditions:

The member has connected to ix-dev, established a connection to the database, and is able to modify/view data. Or the member can modify/view data via Mysql Workbench

Use Case: View location data on webpage

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

Actors: Any user

Preconditions:

1. The user has access to the internet.
2. The user is on a computer.
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 https://ix.cs.uoregon.edu/~masonj/422Project1.html

Postconditions:

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

3. Specific Requirements

The following requirements present the overview of how the system will function with resources and user interaction. External interfaces are listed in order of importance, with the last interface being an option but not pertinent to project 1 so development was not scheduled. Defining the functions relays what information will be passed through each and how it incorporates with the system.

To establish the foundation for a user experience that is positive we will follow usability requirements and several of the performance requirements. The other requirements will develop a system that provides the functionality needed.

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 functionality is based on this data.
3. **Input -** Location data from user cell phones **| Output -** Location data is formatted and stored in our database.
4. User I.D.\tDate\tTime\tLatitude\tLongitude\tTime at Location\n
5. N/A
6. Tab-delimited text

Output Formatted Text Files

1. Output Formatted Text Files
2. This is how raw data can be viewed.
3. **Input -** Location data from database **| Output -** The text file is stored in the project repository.
4. Formatted text from database, output to a '.txt' file.
5. N/A
6. Text file organizing tab-delimited data by User I.D.

SHOULD HAVE:

Backup System

1. Backup Database
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. N/A
6. mySQL database → store in .sql file

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 -** Dots positioned at specific longitudes/latitudes on a Map hosted by Mapbox.
4. A valid .json or .geojson file
5. N/A
6. Website(.html) with a map which connects to the database using a .php file.

COULD HAVE:

User Verification

1. User login
2. This is how a user will receive a user-id, and verify access to the id.
3. **Input -** Username and password **| Output -** Token verifying access
4. Input retrieved from user
5. N/A
6. Token string

WON'T HAVE:

Mobile Visual Representation

1. Visual Representation of Data on a mobile device
2. This enable user to view the data on a device
3. **Input -** mySQL database file **| Output -** visual represent data, e.g. chart, numbers
4. String identifier for visual representation web page.
5. N/A
6. A view on the app that displays our webpage map.

3.2. Functions

- **User Data Collection**
  1. Validity checks on the inputs:
     - Location services accepted, and data provided by device.
  2. Sequence of operations in processing inputs:
     - Obtain current time.
     - If an interval of 5 minutes, obtain necessary data.
     - Insert data to a request body.
  3. 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. To lower the risk of repeated inputs.
    4. Relationship of outputs to inputs, including
        (a) input/output sequence
            (i) Authorize location services.
            (ii) Enable location tracking, pinged every 5 minutes.
            (iii) Data sent in a request to the database webpage.
        (b) formulas for input to output conversion
            (i) Convert coordinates to four decimal places by utilizing rounding and shifting.
            (ii) Obtain formatted dates and times from current day and time via Swift functions.

- **Output Formatted Text Files**
    - 1. Validity checks on the inputs.
        - The result of our query is reformatted using JSON.parse() to ensure it will be usable
    - 2. Sequence of operations in processing inputs.
        - Request data from mysql database
        - Collect data that we want by the query
        - Export text file in tab-delimited text format
    - 3. Responses to abnormal situations, including error handling and recovery.
        - Does not exist in its current form.
    - 4. Relationship of outputs to inputs, including
        - (a) input/output sequences
            - Input: mysql connection and query
            - Output: tab-delimited text files that are written by fwrite in a .php file
        - (b) formulas for input to output conversion
            - mysql connection and query → fwrite function in php file → tab-delimited text file
- **Backup System**
    - 1. Validity checks on the inputs.
        - Mysql server connection is accepted or dropped.
    - 2. Sequence of operations in processing inputs.
        - Get data from mysql server by shell script. Auto-fetch data every 6 hours

        - Store the data in the local host
        - Pushes to GitHub automatically every 6 hours via a shell script.
    - 3. 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.
    - 4. Relationship of outputs to inputs, including
        - (a) input/output sequences
            - (i) 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
            - (ii) Upload to GitHub
                - Input: .sql file
                - Push .sql file to GitHub by script every 6 hours
                - Output: .sql file that is stored on GitHub
        - (b) 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
        - User name: manhimf
        - Host: ix-dev.cs.uoregon.edu
        - Port: 3797
            - CREATE DATABASE CIS422_Project1
            - That is the shell script that to run mysqldump to get data from the database
            - mysqldump -umanhimf -hix-dev.cs.uoregon.edu --port=3797 CIS422_Project1 > $MONTH-$DAY/$HOUR/backup.sql

- Backup_git.sh
    - That is a shell script that used to push backup sql file to github
        - git add .
        - git commit -m "auto push"
        - git push

- Crontab

    - It is a time-based job scheduler in linux
    - Used to schedule and run the script automatically to backup database
        - Backup data every 6 hours
        - Push to github

- Backup.sql
    - Simple sql file that has all data we collected from user identity , time, date, latitude, longitude, time at location.
- 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**
    - 1. Validity checks on the inputs.
        - This data comes directly from our database, so it must be in the proper format already
    - 2. Sequence of operations in processing inputs.
        - Get data file from the 422backend.php page
        - Send its contents to a new HTML div
        - Correct into proper .json format if necessary
        - Data is processed into a forEach loop
        - New HTML divs are created for each datapoint, then those divs are used to create Markers on our map
    - 3. Responses to abnormal situations, including error handling and recovery.
        - The result of our query is reformatted using JSON.parse() to ensure it will be usable
    - 4. Relationship of outputs to inputs, including
        - (a) 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.
        - (b) formulas for input to output conversion
            - Input: SQL query → while($row = $result2->fetch_assoc()){fwrite()...} where we insert datapoints into a manually created json format.

3.3. Usability Requirements

The system will be designed to 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 5 minute interval between recording data, the system will halt most of its actions to reduce resource utilization.

To maintain some privacy a toggle to indicate whether data should be shifted from the home location by a consistent random value between 0.05 and 0.25 miles.

Labeling and organizing the methods in which location tracking is initialized as well as shifting location will increase the user experience when interacting with the application.

Data will be presented in a format that can be interpreted by a reader, following the tab delimited format supplied in the project description. As well as a web page available to see current location.

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

3.4. Performance Requirements

Location data will be collected and posted every 5 minutes.
The system will perform a scheduled backup of the database every 6 hours.
99% of all running applications will send a location post on every 5th minute. Permitting the device's operating system to not execute at the correct tick for a post.
CPU utilization will remain below 5% to retain battery charge for the user.
The text output of data will contain 100% of recorded data.

3.5. Software System Attributes

Reliability:
The system tracks and records location data from many users. To maintain valid data, it is imperative that the system is reliable. To maintain reliability the system will incorporate a backup database and the location data will be stored immediately upon being fetched. For this project, if the tracking location and time are not accurate enough, users would not be able to avoid the crowd, making reliability an important attribute.

Usability:

The usability of the app will be straightforward, as once the user authorizes location tracking, location will be tracked in the background after the push of only 1 button. The only other input by the user will be a button they press when they are at home or leave home, or if they want to stop recording data altogether.

Maintainability:
To preserve portability and maintain a functioning system it is important it be highly maintainable. Organizing functions and components of the system with effective naming conventions and maintaining documentation of system alterations will provide a high level of maintainability. Additionally, all code was commented thoroughly, increasing readability and maintainability down the road.

Safety:
Since the system tracks and saves user location and time, it is important that the data only can be accessed by the development team. Users will have no permission to access or edit the data. Users can only view the visual representation of data. Additionally, to protect the location of the Users homes, they will have the ability to designate when they are at home, which shifts their location by ~.25 miles.

4. References

https://classes.cs.uoregon.edu/20S/cis422/P1/WWeek.pdf
http://www.trailcheck.info/
https://www.mapbox.com/

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

5. Acknowledgements