Software Design Document (SDD)

Team Name: Group 2
Jacob Vasquez, Dominick Daito Jr. , Jolen Reid, Francis Agyemang

Date: December 5, 2024

Contents

Version Description

Version Number	Description	Date Added
1.0	Initial draft	December 5, 2024
2.0	Second draft	December 12, 2024

1 Introduction

1.1 Purpose of the Document

This document is to explain in detail the functions that the application will perform. The document will inform readers as to what the application will do. The purpose of this product is to visualize pedestrian and bicycle data to find and identify problem areas and the safest navigation routes.

1.2 Intended Audience

The main audience of the software requirements specifications document are developers, project managers, and testers. The SRS contains information about each project such as what the project is, what each of its UI elements should do, and what dependencies each project may have. It is suggested that you first look at the table of contents for any topics you may be looking for, if not then quickly skim the document to get a better understanding of the projects. If you are a developer or project manager it is suggested that you look into section 4 of the SRS so that you may check if project requirements are being met. If you are a tester it is suggested that you look into section 3 so that you have a better understanding of how the user interfaces should work.

1.3 Overview of the System

I. Metro Bike Share Real Time (Web and Android App) Angeles area. This is done by modeling real-time Metro Bike Share stations within the city, in conjunction with bike accidents around the city. Currently, the idea is to allow the user to draw the "safest" path by avoiding areas of the city where major accidents have occurred. II. Bicycle Accident Visualization III. Metro Bike Share Historical Data Visualization (Web and Android App) stations are shown as feature layers on the map. Clicking the station displays information such as number of trips and busiest day. Station icons vary in size and color depending on what information we want to show.

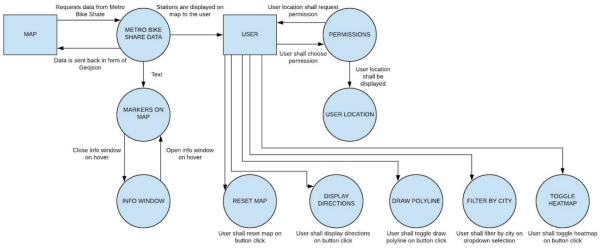
2 System Architecture

2.1 Workflow of the System

Level 0: Data Engines Overview

2.2 Breakdown of Components

- Use of a product(programming language, database, library, etc)



  - Java
  - JavaScript
  - HTML CSS
  - Python
  - Pandas Library
  - SkLearn
  - Google Maps API
  - Android Studio
  - Maps SDK for Android
  - Maps SDK for Android Utility Library
  - Directions API
  - Firebase Database
  - ArcGIS
  - JSON/GeoJSON
  - ArcGIS Runtime SDK

- Reuse of existing software components to implement various parts or features of the system
  - This software is the first version, no reuse of existing software components

- Plans for extending or enhancing the software
  - Collect data in real-time
  - Create a database to hold all data
  - Use Python libraries to manipulate the data

- User Interface paradigms (or system input and output models)
  - Physical mouse required to interact with the application
  - Computer is required to use the application

- Hardware and/or software interface paradigms
  - User's interface will be updated using JavaScript

- Error detection and recovery
  - The error can be checked in the console log in order to see if the data loaded
- External databases and/ or data storage management and persistence
  - Made use of historical data using Geohub
  - Made use of real time data from Metro Bike Share website
  - Firebase
- Management of other resources
  - No other external resources used

3 Design Considerations

Listed are the various issues that need to be addressed before attempting to devise a complete design solution:

1. Metro Bike Share Real Time (Web-App)
   - Dissect the Directions Service object returned by the Maps Javascript API so that we can manipulate it to our liking.
   - Fully understand the Maps Javascript API so that it may be used to its fullest potential
2. Metro Bike Share Real Time (Android-App)
   - Dissect the JSON object returned by the Directions API so that we can manipulate it to our liking.
   - Decoding the encoded polyline paths to plot the polylines on our map.
   - Find an effective way to switch between fragments while remaining in the same activity
3. Bicycle Accident Visualization
4. Metro Bike Share Historical Data Visualization (Web-App)
   - Get the data Decoding the encoded polyline paths to plot the polylines on our map.
   - Find an effective way to switch between fragments while remaining in the same activity
5. Historical data visualization for Metro bike share (Android App)
   - Get the data from the Metro Bike Share Web page and be able to change and manipulate the CSV files to display the data we want to show.
   - Understand the Android ArcGIS Runtime SDK API.

3.1 General Constraints

The list describes the global limitations or constraints that have a significant impact on the design of the system's software:

- Software Environment
  - No Licensed Google Maps API
- End-User Environment
  - Android SDK greater than 16 required.
  - Basic computer inputs and outputs shall be provided by the user such as mouse, keyboard, monitor screen, and desktop.
  - End-users must have valid accounts from Esri ArcGIS before using the application.
- Standards Compliance
  - Data Visualization Engine shall follow the standards-compliance of World Wide Web.
- Interoperability Requirements
  - Data is directly requested from GeoHub or Metro Bike Share Website using JSON
- Data Repository and Distribution Requirements
  - Currently, Liaison has not provided real-time and historical data

3.2 Goals and Guidelines

Listed are the goals, guidelines, and principles that embody the design of the system software:

- The application should strive to achieve Vision Zero's goal
- The application should help visualize data using a map
- The application should make use of historical and real time data.
- The application should help users reach their destination safely.
- Deadline
  - Delivery Date : May 2020
  - Milestone 1
    - Initial Senior Design Meeting
    - Launch Day
    - Meeting Liason
    - Project Requirements

        * Discuss meeting times as a group
        * Assigned team member roles
        * Meeting with advisor
        * Discuss technologies to implement
        * Tutorials in ArcGIS
        * Research Data Visualization on ArcGIS
        * Develop a demo to practice ArcGIS
    - Milestone 2
        * Introduction to Github
            · Make use of GeoHub data to display dataset on a map using ArcGIS
        * Find an alternative to ArcGIS directions API
            · Googles Direction API work with a free limited key
        * Port the ArcGIS web project to the Google Maps API
        * Use metro Bike Share data to visualize pedestrian data
            · Use GeoJSON from Metro's website to help with real time data
        * Design and Develop Android App of MBSRT web-app
            · Utilized Android Studio to begin development
            · Implemented Maps SDK for Android
            · Implemented Firebase Database and Auth into our applications.
    - Milestone 3
        * Incorporate machine learning algorithms to predict bike availability and safest path.
        * Incorporate the use Jupyter Notebook and JavaScript
        * Use python and Sklearn libraries to predict data

4 Architectural Strategies

- Use of product (programming language, database, library, etc.)
    - Java
    - JavaScript
    - HTMl,CSS
    - Python
    - Pandas Library
    - Sklearn Library
    - Google Maps API
    - Android Studio
    - Maps SDK for Android
    - Maps SDK for Android Utility Library
    - Directions AP

  - Firebase Database
  - ArcGIS
  - JSON/GeoJSON
  - ArcGIS Runtime SDK
- Reuse of existing software components to implement various parts or features of the system
  - This software is first version, no reuse of existing software components
- Plans for extending or enhancing the software
  - Collect data in real time
  - Create a database
  - use python libraries.
- User interface paradigms
  - Physical mouse required to interact with the application
  - Computer is required to use the application
- Hardware and/or software interface paradigms
  - User's interface will be updated using JavaScript
- Error detection and recovery
  - The error can be checked in the console log in order to see if the data loaded
- External databases and/ or data storage management and persistence
  - Made use of historical data using Geohub
  - Made use of real time data from Metro Bike Share website
  - Firebase
- Management of Other Resources
  - No other external resources used

5 Policies and Tactics

The tactics used to implement our applications start with accessing APIs and storing the results. Next, these data are processed in a way that is beneficial to building multiple UI elements such as the station markers, dropdown list, and the station list.

5.1 Choice of which specific products used

Esri ArcGIS licenses, Maps Javascript API, Maps SDK for Android, and Python shall be used for visualization of data. Jupyter Notebook, Excel and Maps SDK for Android Utility Library will be used for data processing. Visual Studio Code will be used for bringing together all HTML, CSS and Javascript elements. Android Studio will be used to build and test android components

5.2 Plans for ensuring requirements traceability

The requirements of the application shall be traceable. The source code can be traced via GitHub version control.

5.3 Plans for testing the software

In the future we plan to test various machine learning classifiers. Choosing the classifier that yields the highest accuracy for both bike availability and the safety of a bike route.

6 Detailed System Design

6.1 Data Preprocessing

6.1.1 Responsibilities

The main responsibility of this module is to fetch the data set that will be used for our maps via an API request. This data will then be stored and filtered for specific data.

6.1.2 Constraints

There may be missing features and labels from the preprocessed data because the data is inconsistent and flawed with null data. The data sets could be too large and affect the performance. Larger data sets are also more expensive to process.

6.1.3 Composition

The list below contains widget components for users to interface with the application:

- Zoom in and out to change the map visibility
- Reset map to its original format
- Toggle a heatmap layer on or off
- Filter stations by cities
- Draw polylines on the map based on their safety rating
- Display directions and directions polyline
- Display current location (require GPS to be enabled)
- Reference the Legend
- Select which data layer to be displayed
- Filter by date and time
- Search for a location in the map
- User Profile

6.1.4 Uses/Interaction

Our maps are used to visualize Metro Bike Share stations around the greater Los Angeles area. Our maps also visualize bike accidents that occurred in the greater Los Angeles area. Maps can be used to find stations with available bikes, and allow the user to draw the "safest" path for him/her.

6.1.5 Resources

The following data are used to visualize:

- MetroBikeShare.json
- bikeAccidents.csv
- metroBikeShareData.csv
- interface layout.xml

6.1.6 Interface/Exports

The visualized data is displayed on a map layer created from ArcGIS JavaScript API, Maps Javascript API and Maps SDK for Android.

7 Detailed Lower level Component Design

7.1 Visualization Component

7.1.1 Classification

The component is a visualization map of datasets.

7.1.2 Processing Narrative (PSPEC)

1. Metro Bike Share Real Time (Web-App) Applications get data sets from GeoHub and display them on the map. The user can thenfilter by city or by clicking specific stations. The user can then choose if they want adirect path from their position to their destination, if they allow Google to access theirlocation, or they can choose to draw their own path. Paths displayed between two points will be calculated for a safety rating, then displayed on the map.

2. Metro Bike Share Real Time (Android-App) As for the android application the user will be prompted to sign in or sign up. Once theuser has successfully logged in the map will be visible with its features. The user shall beable to access his/her current location to find the safest possible route to the bike station.Once the application has access to the user's location the stations closest to the user shallbe highlighted blue, in order to show stations within the users radius. The user will alsobe able to click on a filterable list that will contain all the Metro Bike stations availablefor the user to search or click to get the safest possible directions to the station. Uponclicking on a station our algorithm will display 3 best routes for the user to get to thestation using a safety rating based on the bike accidents GeoJson dataset, the directionswill also

be provided. The application will also allow the user to create a custom pathupon clicking the paths button, that will allow the user to choose a starting station andend point station. The application will also provide the user with the ability to have aprofile that will be customizable from the user picture to the user name and other credentials.

7.1.3 Interface Description

Interactive map allowing the user to filter through specific fields selected.

7.1.4 Processing Detail

Manipulating each dataset to extract the fields needed for outputting data onto the map.

7.1.5 Design Class Hierarchy

The visualization component falls below the data pre-processing component.

7.1.6 Restrictions/Limitations

Datasets are too large and have some null fields. Some datasets are not update.

7.1.7 Performance Issues

Station pictures are all the same, does not show actual location

7.1.8 Design Constraints

Can only do as much as what Google Maps API is capable of

7.1.9 Processing Detail For Each Operation

- Geohub data set - filtered for bike accidents only
- Metro Bike Share API Call - destructured object for Geojson object
- City filters - filter markers for only stations within that city

8 User Interface

8.1 How to Use the System

From the user interface, the user can:

- Zoom in and out to change map visibility
- Hovering over the marker displays an info window about the station
- User Location button to prompt the user for their location
- Reset map to its original state using the reset button

- Toggle heat map on and off
- Draw a Polyline to allow the user to draw his/her path by clicking anywhere on the map. This can be toggled on/off
- Click on the station to be highlighted and its corresponding marker will be animated
- Filter by city with the drop-down menu
- Click on a station marker to display directions and a polyline of how to get from the user's location to the clicked station
- Use Google Map's default features

8.2 Database Design and Explanation

No databases are used in our projects for now.

8.3 Screenshots (Optional)

9 Glossary

Acronym	Definition
UI	User Interface
API	Application Programming Interface
DB	Database
ArcGis	Esris all in one solution to work with geographic information
AISC	A.I. for Smart Cities: Pedestrian and Bicycle Safety
CSS	Cascading Style Sheet is a style sheet that is used to describe the presentation of a markup language.
CSV	Comma Separated Values. File format that is used to store tabular data such as spreadsheets or databases.
DFD	Data Flow Diagram.
Firebase	BaaS for cloud storage and authentication. HTML
BaaS for cloud storage and authentication	
HTTP	Hypertext Transfer Protocol is an application protocol for distributed, collaborative, hypermedia information systems.
Javascript	A programming language that is heavily used for web scripts.
LADOT	Los Angeles Department of Transportation.
MBSRT	Metro Bike Share Real Time
Machine learning	Predictive mathematical models are used for predictions.
Operating System	The software allows any computer to communicate, modify, and terminate hardware and software communications based on end-users decisions.

Python	A general-purpose programming language that can also be used to program web applications and data analytics applications.
Runtime	The time when an application is executed.
SDD	Software Design Document.
SRS	Software Requirements Specifications.
SDK	Software Development Kit.

10 References

- ArcGIS All references to ArcGIS services. https://doc.arcgis.com/en/
- GeoHub Data collection for the city of Los Angeles http://geohub.lacity.org/
- Metro Bike Data Anonymized Metro Bike Share trip data for data collection https://bikeshare.met
- The Maps JavaScript API lets you customize maps with your own content and imagery for display on web pages and mobile devices. The Maps JavaScript API features four basic map types (roadmap, satellite,hybrid, and terrain) which you can modify using layers and styles, controls and events, and various services and libraries. https://developers.google.com/maps/documentation/javascript/tutorial
- Android Studio Used to develop MBSRT android version. https://developer.android.com/docs
- Maps SDK for Android. Adds functionality to elements within the map https://developers.google.c sdk/intro
- Google's Directions API Used to retrieve a JSON object containing directions information between points https://developers.google.com/maps/documentation/directions/start
- Maps SDK for Android Utility Library Used to decode the directions polyline from the Directions JSON object. Used to add heatmap layer over map https://developers.google.com/m sdk/utility
- Firebase Used to authenticate and store user data. https://firebase.google.com/
- Jupyter Notebook Organize and manipulate data. https://jupyter.org/