Software Design Document

for

A.I. for Smart Cities: Pedestrian and Bicycle Safety

Version 1.0, approved

Prepared by
Haley Gray, Freddy Gutierrez, Mathew Ramos,
Alejandro Bernal, Marcio Arakaki

City of Los Angeles and LADOT

September 3, 2019
# Table of Contents

- **Table of Contents**.................................................................................................................. pg 2
- **Revision History**.................................................................................................................... pg 2
  - **1. Introduction**............................................................................................................... pg 4
    - 1.1. Purpose............................................................................................................ pg 4
    - 1.2. Document Conventions..................................................................... pg 4
    - 1.3. Intended Audience and Reading Suggestions........................................ pg 4
    - 1.4. System Overview................................................................................ pg 4
  - **2. Design Considerations**.............................................................................................. pg 6
    - 2.1. Assumptions and dependencies........................................................ pg 6
    - 2.2. General Constraints........................................................................ pg 6
    - 2.3. Goals and Guidelines........................................................................ pg 7
    - 2.4. Development Methods...................................................................... pg 8
  - **3. Architectural Strategies**........................................................................................... pg 9
  - **4. System Architecture**................................................................................................ pg 10
  - **5. Policies and Tactics**................................................................................................. pg 11
    - 5.1. Specific Products Used........................................................................ pg 11
    - 5.2. Requirements traceability.............................................................. pg 11
    - 5.3. Testing the software........................................................................ pg 11
  - **6. Detailed System Design**.......................................................................................... pg 10
    - 6.1 Data Preprocessing................................................................................ pg 10
      - 6.1.1 Responsibilities............................................................................... pg 10
      - 6.1.2 Constraints.................................................................................... pg 10
      - 6.1.3 Composition................................................................................ pg 10
      - 6.1.4 Uses/Interactions........................................................................ pg 10
      - 6.1.5 Resources..................................................................................... pg 10
      - 6.1.6 Interface/Exports.......................................................................... pg 10
    - 6.1 DataVisualization.......................................................................................... pg 13
      - 6.2.1 Responsibilities............................................................................... pg 13
      - 6.2.2 Constraints.................................................................................... pg 13
      - 6.2.3 Composition................................................................................ pg 13
      - 6.2.4 Uses/Interactions........................................................................ pg 13
      - 6.2.5 Resources..................................................................................... pg 14
      - 6.2.6 Interface/Exports.......................................................................... pg 14
  - **7. Detailed Lower level Component Design**................................................................ pg 15
    - 7.1 Visualization Component........................................................................ pg 15
      - 7.1.1 Classification................................................................................. pg 15
      - 7.1.2 Processing Narrative(PSPEC)...................................................... pg 15
      - 7.1.3 Interface Description................................................................ pg 15
      - 7.1.4 Processing Detail........................................................................ pg 15
        - 7.1.4.1 Design Class Hierarchy......................................................... pg 15
        - 7.1.4.2 Restrictions/Limitations...................................................... pg 15
        - 7.1.4.3 Performance Issues.............................................................. pg 15
        - 7.1.4.4 Design Constraints.............................................................. pg 15
        - 7.1.4.5 Processing Detail For Each Operation.............................. pg 16
  - **8. Database Design**....................................................................................................... pg 17
9. User Interface............................................................................................................pg 18
   9.1. Overview of User Interface......................................................................... pg 18
   9.2. Screen Frameworks or Images................................................................. pg 18
   9.3. User Interface Flow Model.........................................................................pg 20
10. Requirements Validation and Verification......................................................... pg 21
11. Glossary................................................................................................................... pg 24
12. References.............................................................................................................pg 25

Revision History

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Reason For Changes</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freddy, Marcio, Matthew, Alejandro, Haley</td>
<td>12/08/19</td>
<td>Revised sections 1, 2, 3, 4, 5, Appendix</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Purpose

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 identified problem areas and safety routes.

1.2 Document Conventions

Roman numerals indicates each application. Bullet points are used to describe the applications. External links will be underlined in blue.

1.3 Intended Audience and Reading Suggestions

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.

1.4 System Overview

1. Metro Bike Share Real Time
   • The focus of this project is to aid in the realization of Vision 0 is the greater Los 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
2. Design Considerations

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

I. Metro Bike Share Real Time
   ● 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

II. Bicycle Accident Visualization

III. Metro Bike Share Historical Data Visualization

2.1 Assumptions and Dependencies

I. Metro Bike Share Real Time
   ● User accepts location permission when prompted
   ● Metro Bike Share keeps their real time data public, and in the same format
   ● Developers are provided with a Google Maps API key that has access to required APIs

II. Bicycle Accident Visualization
   ● Los Angeles Geohub keeps the data set available and in the same format.

III. Metro Bike Share Historical Data Visualization
   ● Metro Bike Share keeps their keeps their historical data public and in the same format

2.2 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**
  - A browser is required to view the visualizations.
  - Basic computer inputs and outputs shall be provided by the user such as mouse, keyboard, monitor screen, and desktop.
  - End-user must have valid accounts from Esri ArcGIS prior to 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**
  - As of now, real time and historical data have not been provided by Liaison.

### 2.3 Goals and Guidelines

Listed are the goals, guidelines and principles which embody the design of the system’s software:

- The application should strive to achieve Vision Zero’s goal.
- The application should help visualize data using on 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 Liaison
    - Project Requirements
    - Discussed meeting times as a groups
    - Assigned Team roles
    - Meeting with advisor
■ Discuss technologies to implement
■ Tutorials in ArcGIS
■ Research data visualization using ArcGIS
■ Develop a demo to practice ArcGIS further
  ○ Milestone 2
    ■ Introduction to GeoHub
      ● Make use of GeoHub data to display dataset on a map using ArcGIS
    ■ Find an alternative to ArcGIS directions API
      ● Required License
    ■ Use Google Maps API
      ● Google’s Direction API works 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
  ○ 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
  ○ Final Delivery Date: TBA

2.4 Development Methods
3. Architectural Strategies

- **Use of a product (programming language, database, library, etc)**
  - JavaScript
  - HTML, CSS
  - Python
  - Pandas Library
  - Sklearn Library
  - Google Maps API
  - ArcGIS

- **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 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

- **Management of other resources**
  - No other external resources used
4. System Architecture

Level 0: Data Engines Overview

Level 1: Engine Process Flow Over
5. Policies and Tactics

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

5.1 Choice of which specific products used

Esri ArcGIS licenses, Maps Javascript API, and Python shall be used for visualization of data. Jupyter Notebook and Excel will be used for data processing. Visual Studio Code will be used for bringing together all HTML, CSS and Javascript elements.

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

- MetroBikeShare.json - geojson object returned from API call to Metro Bike Share site
- bikeAccidents.csv - filtered GeoHub csv file for only bike accidents
- metroBikeShareData.csv - filtered the csv file for stations, location, day, and time

6.1.4 Uses/Interactions

The data files that are preprocessed by this module are then used in module 6.2 for data visualization.

6.1.5 Resources

The following data are used to visualize:

- MetroBikeShare.json
- bikeAccidents.csv
- metroBikeShareData.csv

6.1.6 Interface/Exports

The visualized data is displayed on a map layer created from ArcGIS JavaScript API.
6.2 Data Visualization

6.2.1 Responsibilities
Take the preprocessed data from module 6.1 and translate into several different UI components on the map, such as station markers, station lists, city filters, etc. Also create a layer of polygons that represent stations. The bigger the polygon, the more demanding bikes at that station are. The more opaque the polygon the later in the day it is.

6.2.2 Constraints
The output is limited because not all data may be displayed on the map. Some datasets are not as recent as others which is not ideal for our situation. Also the inconsistencies of null data fields could cause data not to show up on the map.

6.2.3 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
- 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

6.2.4 Uses/Interactions
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 “safer” path for him/her.

6.2.5 Resources
The following data are used to visualize:

- MetroBikeShare.json
- bikeAccidents.csv
- metroBikeShareData.csv

6.2.6 Interface/Exports

The visualized data is displayed on a map layer created from ArcGIS JavaScript API and Maps Javascript API.
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)

Application get data sets from GeoHub and displays them on the map. The user can then filter by city or by clicking specific stations. The user can then choose if they want a direct path from their position to their destination, if they allow Google to access their location, or they can choose to draw their own path.

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.

7.1.4.1 Design Class Hierarchy

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

7.1.4.2 Restrictions/Limitations

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

7.1.4.3 Performance Issues

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

7.1.4.4 Design Constraints
Can only do as much as what Google Maps API is capable of.

7.1.4.5 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. Database Design

No databases are used in our projects for now.
9. User Interface

9.1 Overview of User Interface

From the user interface, the user can:

- Zoom in and out to change map visibility
- Hovering over 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 toggle on/off.
- Click on 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 defaults features

9.2 Screen Frameworks or Images
9.3 User Interface Flow Model
10. Requirements Validation and Verification

I. Metro Bike Share Real Time Module

<table>
<thead>
<tr>
<th></th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1</td>
<td>This system shall fetch a Geojson object from the Metro Bike Share real time data set, and display a marker on the map for each station.</td>
</tr>
<tr>
<td>4.1.2</td>
<td>This system shall display an info window on the hover of a station.</td>
</tr>
<tr>
<td>4.1.3</td>
<td>This system shall remove an info window when hovering off a station.</td>
</tr>
<tr>
<td>4.1.4</td>
<td>The system shall prompt the user for his/her location when the “User Location” button is clicked</td>
</tr>
<tr>
<td>4.1.5</td>
<td>The system shall reset the map to its original state when the “Reset Map” button is clicked</td>
</tr>
<tr>
<td>4.1.6</td>
<td>The system shall toggle a heatmap representative of each accident in the Geohub data set when the “Toggle Heatmap” button is clicked</td>
</tr>
<tr>
<td>4.1.7</td>
<td>The system shall draw a polyline on the map when the “Draw Polyline” button is toggled on, and the map is clicked</td>
</tr>
<tr>
<td>4.1.8</td>
<td>The system shall highlight a station item and move the map to the corresponding marker on the map when the station item is clicked</td>
</tr>
<tr>
<td>4.1.9</td>
<td>The system shall display only markers on the map which have the same value as the “Filter By City” dropdown.</td>
</tr>
<tr>
<td>4.2.0</td>
<td>The system shall display directions and a polyline of how to get from the user’s location to the clicked station</td>
</tr>
</tbody>
</table>

II. Bicycle Accident Visualization Module

<table>
<thead>
<tr>
<th></th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1</td>
<td>The system shall use the Feature Layer Service provided by Los Angeles Geohub and the dataset, Los_Angeles_Collisions_2012through2018, to retrieve bicycle accident data.</td>
</tr>
<tr>
<td>4.1.2</td>
<td>The system shall use the accident data to display markers on a map of Los Angeles.</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Accident severity shall be visualized using colors from yellow and increasing to red to indicate increasing severity.</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Accident severity shall be visualized using increasing marker size for the number of injuries.</td>
</tr>
<tr>
<td>4.1.5</td>
<td>Accident information including year and time shall be displayed in a popup.</td>
</tr>
<tr>
<td>4.1.6</td>
<td>The system shall use the Map Layer Service provided by Los Angeles Geohub and the dataset, Bikeways (Existing), to retrieve bikeway data.</td>
</tr>
<tr>
<td>4.1.7</td>
<td>The system shall use bikeway data to display bikeways on a map of Los Angeles.</td>
</tr>
<tr>
<td>4.1.8</td>
<td>The system shall use different colors to distinguish types of bikeways.</td>
</tr>
<tr>
<td>4.1.9</td>
<td>The system shall use the Map Layer Service provided by Los Angeles Geohub and the dataset, 2000 Census Tracts, to retrieve polygons.</td>
</tr>
<tr>
<td>4.2.0</td>
<td>The system shall use the polygon data to display the polygons on the map.</td>
</tr>
<tr>
<td>4.2.1</td>
<td>The system shall provide counts of bicycle accidents within the area of the polygons in a popup.</td>
</tr>
<tr>
<td>4.2.2</td>
<td>The system should have a function to display accidents within a radius of a mouse click.</td>
</tr>
<tr>
<td>4.2.2</td>
<td>The system should allow the user to change the radius for displaying accidents.</td>
</tr>
<tr>
<td>4.2.3</td>
<td>The system should allow for filtering of bicycle accidents displayed on the map by year.</td>
</tr>
<tr>
<td>4.2.4</td>
<td>The system should allow for filtering of bikeways displayed on the map by type.</td>
</tr>
</tbody>
</table>

### III. Metro Bike Share Historical Data Visualization
<table>
<thead>
<tr>
<th>4.1.1</th>
<th>The system shall use the Feature Layer Service provided by ArcGIS Online services and the dataset, Metro Bike Share Historical data, to retrieve trip history data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.2</td>
<td>The system shall use the station location to display markers on a map of Los Angeles.</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Metro bike share stations peak hours shall be visualized using colors and opacity features</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Metro bike share stations popularity shall be visualized by the size of the polygon marker.</td>
</tr>
<tr>
<td>4.1.5</td>
<td>The system shall use the polygon data to display the polygons on the map.</td>
</tr>
<tr>
<td>4.1.6</td>
<td>The system should have a function to display a pop-up window of trip statistics.</td>
</tr>
<tr>
<td>4.1.7</td>
<td>The system should allow the user to filter through the days of the week for more visualization of each station.</td>
</tr>
</tbody>
</table>
## 11. Glossary

<table>
<thead>
<tr>
<th><strong>API</strong></th>
<th>Application program interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArcGis</td>
<td>Esri’s all-in-one solution to work with geographic information.</td>
</tr>
<tr>
<td>AISC</td>
<td>A.I. for Smart Cities: Pedestrian and Bicycle Safety</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheet is a style sheet that is used to describe the</td>
</tr>
<tr>
<td></td>
<td>presentation of a markup language.</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma Separated Values. File format that is used to store tabular</td>
</tr>
<tr>
<td></td>
<td>data such as spreadsheets or databases.</td>
</tr>
<tr>
<td>DFD</td>
<td>Data Flow Diagram</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language is the standard markup language for</td>
</tr>
<tr>
<td></td>
<td>creating web pages.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol is an application protocol for</td>
</tr>
<tr>
<td></td>
<td>distributed, collaborative, hypermedia information systems</td>
</tr>
<tr>
<td>Javascript</td>
<td>A programming language that is heavily used for web applications</td>
</tr>
<tr>
<td>LADOT</td>
<td>Los Angeles Department of Transportation</td>
</tr>
<tr>
<td>MBSRT</td>
<td>Metro Bike Share Real Time</td>
</tr>
<tr>
<td>Machine</td>
<td>Predictive mathematical model used for predictions</td>
</tr>
<tr>
<td>learning</td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>The software that allows any computer to communicate, modify, and</td>
</tr>
<tr>
<td>System</td>
<td>terminate any hardware and software communications based on end-users</td>
</tr>
<tr>
<td></td>
<td>decisions.</td>
</tr>
<tr>
<td>Python</td>
<td>A general-purpose programming language that can also be used to</td>
</tr>
<tr>
<td></td>
<td>program web application and data analytics application.</td>
</tr>
<tr>
<td>Runtime</td>
<td>The time when an application is executed.</td>
</tr>
<tr>
<td>SDD</td>
<td>Software Design Document</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirements Specifications</td>
</tr>
</tbody>
</table>
## 12. References

<table>
<thead>
<tr>
<th>Alias</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArcGIS</td>
<td>All references to ArcGIS services. <a href="https://doc.arcgis.com/en/">https://doc.arcgis.com/en/</a></td>
</tr>
<tr>
<td>Metro Bike Data</td>
<td>Anonymized Metro Bike Share trip data for data collection <a href="https://bikeshare.metro.net/about/data/">https://bikeshare.metro.net/about/data/</a></td>
</tr>
<tr>
<td>Google Maps API</td>
<td>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. <a href="https://developers.google.com/maps/documentation/javascript/tutorial">https://developers.google.com/maps/documentation/javascript/tutorial</a></td>
</tr>
<tr>
<td>Jupyter Notebook</td>
<td>Organize and manipulate data.</td>
</tr>
</tbody>
</table>