

# **Software Design Document (SDD)**

**for**

## **Sidewalk Slope Assessment System**

**Version 2**

**Prepared by:**

Aquil Alam, Alejandro Chanocua, Omar Eclicerio, Ernesto Garcia, Francisco Gastelum, Henry Gonzales, Gui He, Perla Ramirez, Rishi Shah, Daniel Zeng

**Department of Public Works and Bureau of Engineering, LA City**

**CSULA Senior Design 2021-2022 / LA City Bureau of Engineering**

# Table of Contents

<b>1.0 Introduction</b>	<b>5</b>
1.1 Purpose	5
1.2 Document Conventions	5
1.3 Intended Audience and Reading Suggestions	5
1.4 System Overview	5
<b>2.0 Design Considerations</b>	<b>7</b>
2.1 Assumptions and Dependencies	7
2.2 General Constraints	8
2.3 Goals and Guidelines	9
2.4 Development Methods	10
<b>3.0 Architectural Strategies</b>	<b>12</b>
3.1 Task 1	12
3.2 Task 4	13
<b>4.0 System Architecture</b>	<b>14</b>
4.1 Task 1	14
4.2 Task 2	15
4.2.1 Rover UI	15
4.3 Task 4	16
4.3.1 Database Data Diagram	16
4.3.2 Sidewalk Table	17
4.3.3 Rover data table	20
4.3.4 Rover sidewalk	22
4.3.5 Sidewalk GoPro Metadata	24
4.3.6 Sidewalk Rover Data View	26
<b>5.0 Policies and Tactics</b>	<b>30</b>
5.1 Choice of which specific products used	30
5.2 Plans for ensuring requirements traceability	31
5.3 Plans for testing the software	31
<b>6.0 Detailed System Design</b>	<b>32</b>
6.1 Raspberry Pi 3	32
6.1.1 Responsibilities	32
6.1.2 Constraints	32
6.1.3 Composition	32
6.1.4 Resources	32
6.1.5 Interface/Exports	32
6.2 Rover User Interface	32
6.2.1 Responsibilities	32
6.2.2 Constraints	32
6.2.3 Composition	32
6.2.4 Uses/Interactions	32
6.2.5 Resources	32
6.3 NavigateLA	33

6.3.1 Responsibilities	33
6.3.2 Constraints	33
6.3.3 Composition	33
6.3.4 Uses/Interactions	33
6.3.5 Resources	33
6.3.6 Interface/Exports	33
6.4 Web application and mapping files	33
6.4.1 Responsibilities	33
6.4.2 Constraints	33
6.4.3 Composition	34
6.4.4 Uses/Interactions	34
6.4.5 Resources	34
6.4.6 Interface/Exports	34
<b>7.0 Rover UI Design</b>	<b>35</b>
7.1 Rover UI Input Field and Buttons	35
7.2 Rover UI JoyStick	35
<b>8.0 Database Design</b>	<b>36</b>
8.1 Relational Schema	36
8.2 Diagram Description	36
8.3 Azure blob Storage implementation	36
8.4 Data Collection	37
8.5 Future Implementation	37
<b>9.0 User Interface</b>	<b>38</b>
9.1 Overview of User Interface	38
9.1.1 Task 1, Web Application	38
9.2 Screen Frameworks or Images	38
9.2.1 Task 1, Web Application	38
9.2.2 Task 3, Rover UI	42
9.3 User Interface Flow Model	43
9.3.1 Task 1, Web Application	43
<b>10.0 Requirements Validation and Verification</b>	<b>44</b>
<b>11.0 Glossary</b>	<b>45</b>
<b>12.0 References</b>	<b>45</b>

## Revision History

Name	Date	Reason For Changes	Version
Aquil Alam	5/10/2022	Spring 2022 Draft Prepare	2
Perla Ramirez	5/11/2022	Added descriptions to sections 1 and 2	2
Perla Ramirez, Ernesto Garcia Alejandro Chanocua, Aquil Alam	5/11/2022	Fixed formatting for headers, changed table of contents for automatic updates. Updated descriptions, contents/explanations for Sections contributed by each person, details on project elaborated.	2
Daniel Zeng, Omar Eclicerio	5/11/2022	Modified sections 2.3,2.4, 4, 5.1, 6.3, and updated the table of contents. Content updated.	2
Francisco Gastelum, Gui He, Rishi Shah	5/11/2022	Modified sections 2, 3, 4, 5, 6 from a task 1 perspective.	2
Henry Gonzales	5/11/2022	Updated task 1 for each section	2

# 1.0 Introduction

## 1.1 Purpose

The Sidewalk Slope Monitoring System project is an effort to develop the necessary databases, user interfaces, and automation scripts to aid the City of Los Angeles, Bureau of Engineering (BOE) in maintaining over 11,000 miles of sidewalk. Based on their prioritization and scoring system, BOE can assign a numerical score to each sidewalk segment to determine which segments require immediate attention or repair for their Sidewalk Repair Program.

The system developed by our team is designed to help BOE in their data collection by building a robot user interface to control the robot BOE uses during their field analysis, image processing scripts to extract image information, a web application to display and map image data, and a database to hold the data collected and extracted by the system.

This document will outline the design of the system explained above. See [System Overview](#) for an exact breakdown of the system into tasks.

## 1.2 Document Conventions

This Software Design Document (SDD) uses a standard documentation template provided by the California State University, Los Angeles Computer Science department.

## 1.3 Intended Audience and Reading Suggestions

This document is intended to aid the team members listed on the first page in scoping and developing their initial software design requirements that will later be refined in the Software Requirements Specification (SRS) document. This document is intended to break down BOE's initial project requirements into tasks to aid the team in planning work. The entire document should be read by the team. Although team members are ultimately held responsible for their specific tasks, team members should be aware of the entire team's efforts.

This document is intended to provide BOE and the project advisor with an overview of the expected work to be completed by the team. Sections 2, 6, and 10 should be read by BOE and the project advisor.

This document is intended to provide a high-level overview of design planning and concepts for Computer Science students involved in future years' efforts. Sections 2, 3, 4, 6, 9, and 10 should be read by Computer Science students who will continue this project beyond Spring 2022.

## 1.4 System Overview

BOE requires sidewalk data to evaluate sidewalk segments for their Sidewalk Repair Program. Because there are over 11,000 miles of sidewalks, it is valuable to BOE to leverage a process that minimizes overhead by automating image and data processing.

To collect the necessary data to assign scores, BOE plans to utilize a Leo Rover robot, with a

GoPro camera mounted on it, to collect images of sidewalk segments in the city of Los Angeles. The system as a whole is broken down into four major components or tasks. The process for BOE to use our system is as follows:

1. The robot will be accompanied by a field worker that will control it using a user interface developed by the team. The user interface is hereby referenced to as Task 3.
2. Scripts will be produced in an effort to parse Exchangeable Image File (EXIF) data from the GoPro images collected. Image processing scripts will be applied to the images to extract additional data necessary for BOE's prioritization and scoring system. The image processing scripts will hereby be referenced as Task 2 and begin in the Spring 2022 semester.
3. A web application will display the collected images with its related EXIF and image processing data according to latitude and longitude. In addition, Task 1 will develop scripts to create mapping files that can be hosted on BOE's mapping application, NavigateLA. The web application and the mapping files will hereby be referenced as Task 1.
4. A database developed by the team will contain tables for the processed GoPro EXIF data, rover data, and GeoJSON ready shapes based on collected sidewalk data. The database is hereby referenced as Task 4.

## 2.0 Design Considerations

### 2.1 Assumptions and Dependencies

- Rover expectations
  - Cracks and holes on sidewalks shall be minimal.
  - Battery 4 hrs of nominal driving or 8 hrs of video streaming.
  - Hardware is reliable.
  - Users will protect hardware from damage.
  - System is waterproof.
  - Operator will clear the sidewalk before measurement.
- Task 1
  - Web application
    - Web application will be used to view specific or individual sidewalk segment images and slope data.
    - Web application will be used to visually review sidewalk segments.
    - Web application will be used to visualize and package collected Rover data for NavigateLA.
  - Mapping files
    - Mapping files will be viewed by the user by importing them to NavigateLA, if and only if, mapping files are not hosted on NavigateLA.
    - Mapping files will contain data for many sidewalk segments.
    - Mapping files will be viewed by the user by selecting a layer loaded on NavigateLA, if and only if, mapping files are kept on a database managed by BOE.
    - Mapping files will contain its damage categorization based on the Severity Index from the Damage Severity Matrix, provided by the BOE report.
    - Mapping files will contain all other information that is important for the BOE to view on NavigateLA.
- Task 2
  - System will be used during the day to take optimal photos.
  - Sidewalk will always be captured as the center of the image
  - Rendered images requires user-input before processing measurements
- Task 4
  - Rover
    - User will use GoPro provided app to render 360 Degree Images
    - User will keep an eye on the rover at all times.
    - User will manually extract data from rover's memory cards

- User will manually control the Rover using the Rover UI.
- Web API
  - User requests data through URL or http requests.
  - Users will only upload unmodified CSV files to the CSV upload route.
  - User requests database queries in JSON or JSON encoded URI format.
- Database
  - Users will manually input data into the database or through API endpoints.
  - User specifies either a sidewalk or section identifier for every rover scan.

## 2.2 General Constraints

- Task 1
  - Web application
    - Web application uses a local instance for development and testing purposes.
    - Web application must have access to the database containing sidewalk data.
    - Web application must use database when displaying sidewalk data and cannot store or use data locally.
    - Web application must use Azure Storage Blob to access GoPro images.
    - Web application must use a BOE web server for production.
    - Web application must be accessible to BOE, either using Internet or Intranet access.
    - Web application must be accessible to BOE regardless of their preferred browser and screen size.
    - BOE must provision a web server for web application production use.
    - BOE must maintain the web server hosting the web application.
    - Web application backend will be done using React.
    - Web application frontend will be done using HTML/CSS and Bootstrap.
  - Mapping files
    - Mapping files must be generated by scripts using a 3<sup>rd</sup> party application like ArcGIS.
    - Mapping files must be in either: CSV, DXF, DWG, DGN, KML, geoJSON, or shapefile formats in order to be imported on NavigateLA.
    - Mapping files must be importable in NavigateLA, if and only if, mapping files will not be hosted on a database managed by BOE.
    - Mapping files must maintain all design features, such as color schemes, annotations, polygons, lines, shapes, when hosted in NavigateLA.
    - Mapping files must be hosted on NavigateLA when a substantial amount of mapping files is available for viewing.



- A request for a database to host the mapping files can only be placed after discussing and receiving approval from the advisor and BOE.
- Task 3
  - Database
    - Access to BOE Sidewalk database must be granted by BOE
    - IP address (assigned by ISP) must be manually approved to access database
  - Web API Server
    - Web host is required for deployment.
    - Operating system required.
    - NodeJS installation required.
    - IP of API server must be approved by BOE to access database
    - Manual handling of the .env file containing credentials is needed for API to connect to the database.
    - Ports on the deployment server must be opened in accordance with the port selected in the API code.
    - Python installation is required to use CSV upload features from API.
    - Temporarily created directories storing CSV files are to be deleted after a set period to prevent unnecessary storage waste.

## 2.3 Goals and Guidelines

- Task 1
  - Web application
    - Web application will be hosted on a web server when the frontend and major components of the backend are complete.
    - Web application's web server will be discussed with BOE to determine what languages and web server applications are available to us.
  - Mapping files
    - Mapping files will be hosted on NavigateLA after receiving approval from advisor and BOE to request the database. Database will host files that NavigateLA will reference. Users can select the files as a layer on NavigateLA.
    - Mapping files will be available for users to import to NavigateLA if a database is not available or if only a limited section of the city is mapped by the mapping files.
    - Mapping files will be packaged by the Date the Field test was conducted with the Rover.
    - Mapping files will contain sidewalk damage categorization based on the percent cross slope from the digital level.

- Task 2
  - Rover
    - Build Leo Rover
    - Implement last year's project onto the Leo Rover.
    - Implement this year's tasks on the Leo Rover
- Task 3
  - Database
    - Design a database schema which will include all aspects of the project. Including but not limited to: rover, UI's, and NavLA.
    - For data that has been preprocessed before insertion, a copy of the unprocessed data should be stored. This makes it possible when changes to preprocessing methods are needed or corrections are made.
    - Collected data should have an association to larger entities such as SectionID or SidewalkID.
    - Implement Azure tools to help with data manipulation.

## 2.4 Development Methods

Components to build a slope monitoring system have been divided into four applicable tasks for early stages of the system. Development methods are yet to be discussed.

- Task 1
  - Web application
    - Use prior knowledge from web development courses and projects to build the frontend and backend of the site.
    - Refer to ReactJS documentation for best practices and setting up an environment.
    - Refer to the Google Maps React API documentation for setting up an interactive map on the web application.
  - Mapping files
    - Understanding ArcGIS and developing methods to map rover data and apply color coding based on Severity Index.
    - Reference ArcGIS forums to explore other user's methods and processes when developing scripts using ArcGIS.
- Task 2
  - Rover User Interface allows field users to control the rover and collect sidewalk data.
- Task 3
  - Database

- Microsoft SQL Server Management Studio 18 software is used to create, edit, delete, and visualize tables.
  - Output files from rover and GoPro are used as guidelines for design and implementation of tables.
  - All text based collected and generated data are designated to be stored in their respective tables.
- o API Server
- A DigitalOcean droplet is used to host the web server containing the API. DigitalOcean was considered due to the free student credits, ease of setting up, and availability of Ubuntu. The choice of web host is independent of the API code, but a migration to a Microsoft based web host is recommended for consistency with other data sources of the client.
  - Ubuntu is the operating system used for the web server. Ubuntu was chosen as the most financially feasible for development and JavaScript is OS agnostic.
  - NodeJS was chosen in place of Python to run the code to remain consistent with the front end team's usage of JavaScript.
  - ExpressJS library is used as the back end web application framework to develop the API. ExpressJS is widely supported and is known to be the de facto framework for back end development with NodeJS.
  - MS-SQL library is used to connect and send commands to the Microsoft SQL database from the API server. MS-SQL provides all the necessary functionality needed and compliance for the API and is supported by Microsoft.
  - API follows standard REST design philosophies in its implementation. REST is commonly used and serves as a solid guideline for further development of API features.
  - CSV uploads to the API server is initially handled by the designated API endpoint and then a child process is spawned to run a Python script to process write to the server.

## 3.0 Architectural Strategies

### 3.1 Task 1

- Web application
  - Used React to combine HTML, CSS, Bootstrap for the frontend and Node.js for the backend. Javascript was used for both the frontend and backend.
    - Enables team to utilize each other's scripts and combine efforts without being concerned with language compatibility.
    - React uses templates that make it easier in developing the backend.
    - React testing environments can easily be created and can easily be replicated following the ReactJS documentation.
    - ReactJS and Node.js have an extensive set of documentation that we have heavily depended on to ensure our product is created with best practices.
  - Python used to create scripts to parse EXIF data from GoPro images
  - React was used to create scripts that extract database data and to extract images from Azure Storage Blob.
  - Google Maps React API was used to create an interactive map
  - GoPro Fusion Studio was used to manually convert all dual fisheye images into equirectangular images with 360 capabilities.
- Mapping files
  - Rover data was loaded onto CSV since NavigateLA web map supports that format.
  - React-csv library used for saving CSV files from the web application's NavigateLA page.
  - Other mapping applications such as ArcGIS are available and their file types are supported by NavigateLA.
    - ArcGIS is also available to students for free from the university, but is only accessible via Parallels Client, a remote desktop application that allows users to access campus servers containing the application. Because we are remotely accessing the server from our personal machines, performance is limited and requires an internet connection.
    - ArcGIS is also used by BOE.
  - Once many mapping files are created to cover a large portion of the city of Los Angeles, the mapping files can be hosted on NavigateLA. This is the next step of the project for FALL2022.
    - Users would be able to select our layers on NavigateLA.

- Otherwise, users would need to import the mapping files and apply color schemes. When imported to NavigateLA, color schemes are not maintained. Any colors or annotations applied to the mapping file are forgotten.
- However, in order for the layer to be hosted on NavigateLA, large amounts of sidewalk data and fully functioning automation scripts are required.

### 3.2 Task 3

- Database

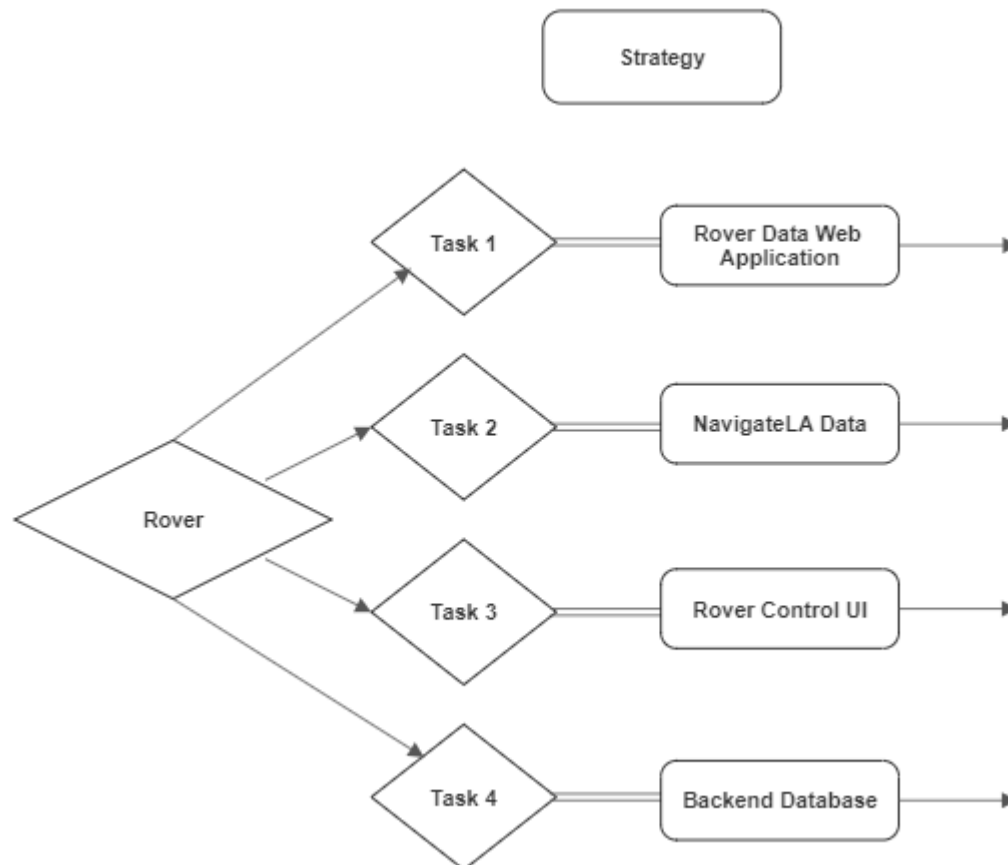
- Collected raw data and processed data are both inserted into the database either in the same table or tables that are associated. This allows adjustments and corrections to the data, should data process methods change.
- All collected data entries from rover and GoPro have associations with larger entities such as sidewalks or street sections to allow easier grouping of data. Using section identifiers is preferred as they are less likely to be changed as opposed to sidewalks.

- Web API Server

- NodeJS allows the deployment of the API server on any platform and requires minimal setup to get running. NodeJS has long term support, wide adoption, and performance features to allow scaling of the applications to millions of requests provided the web provider is sufficient. The OS agnostic attribute of NodeJS provides wide options for migrations from server to server or multiple deployments.
- ExpressJS library offers all the necessary features to implement a REST API. This allows CRUD features to be implemented without complexity and middleware can be applied to adjust server needs.
- All routes are modular by design to promote separation of concerns. Each route handler function contains at most two parameters, one is the database client and the other contains URI parameters and request body if the request type is POST. This implementation allows each route module to execute commands to the database and de-couples from other routes for easier debugging and maintenance.
- Database query route uses JSON object or JSON object encoded to URI to generate SQL statements. The JSON structure allows for customization of queries to the database without directly writing SQL. This reduces the risk of exposing the ability of writing SQL statements directly and offers many of the organization and filtering features of SQL statements. The interchangeability of using either JSON or URI allows for data requests to be completed through either a browser or http request.
- CSV file handling is processed using Python scripts that are executed through a child process spawn from NodeJS. Although this adds complexity of two languages, the main focus of the Python side is on processing data from files and writing to the appropriate tables on the database. CSV processing is far more intuitive with pandas library than offerings available on the JavaScript side and is easier to adapt to changing data inputs in Python. The API's focus is primarily on accessibility of data and less on the processing of data.

## 4.0 System Architecture

The main functionalities of the system are divided into four tasks, then brought together at the end. The four tasks; create a web application for rover data, create a user interface for rover controls, and create a backend database.



### 4.1 Task 1

#### ■ Web Application

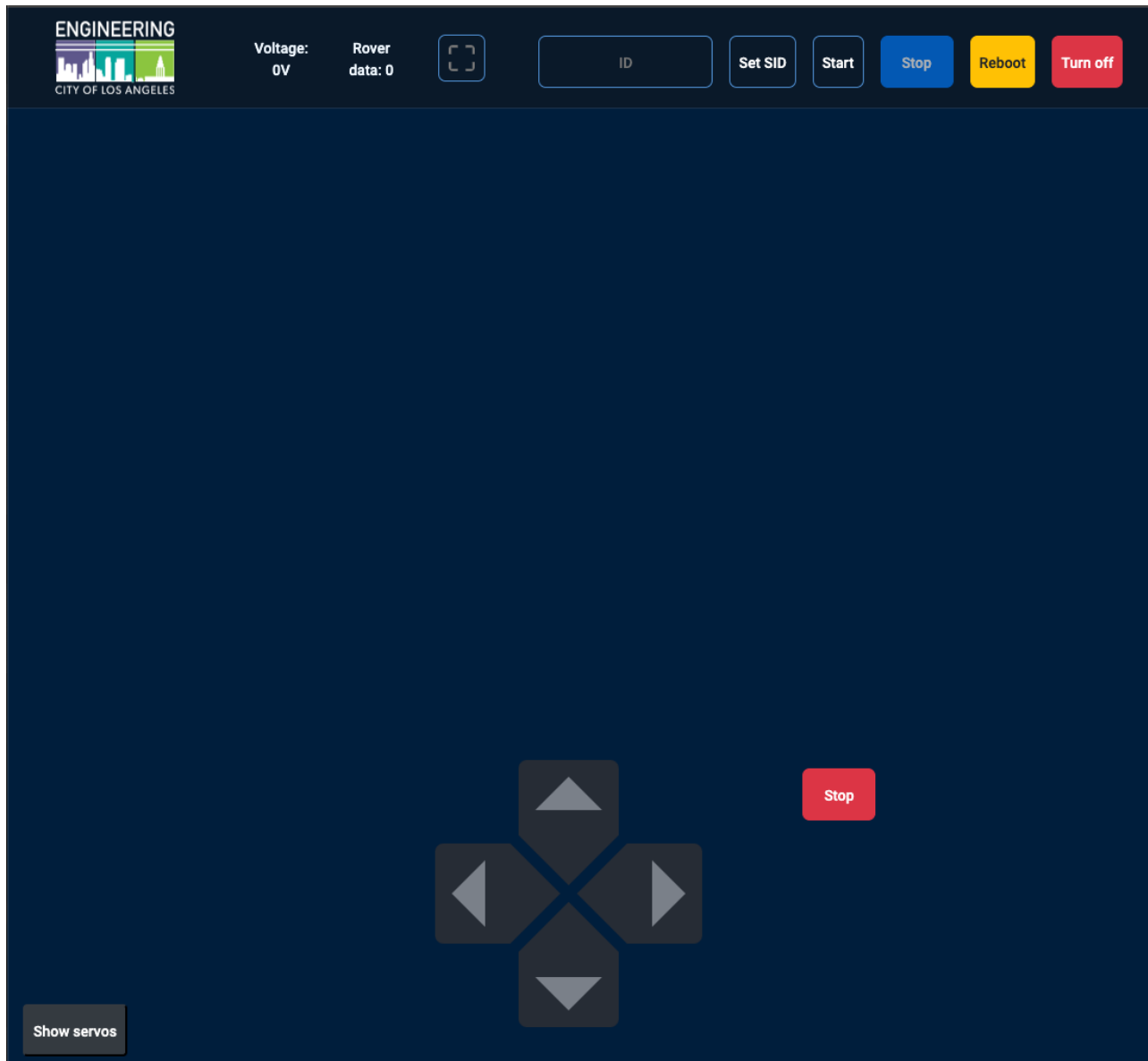
- The web application will display a GoPro image and all of its related data on the console. This includes GPS data, slope data, and other data. Users can switch back and forth between images using “previous” and “next” buttons.
- The web application is intended to show one instance of a sidewalk segment at a time. It will not provide an overview of the entire city of Los Angeles.
- Data shown on the web console is obtained from the Azure storage and SQL Server database. The console contains data from the image processing and database teams.
- The web application’s NavigateLA page will show all of the Rover’s collected data from the database. It can be downloaded as a csv file allowing for its packing for NavigateLA map visualization.

## ■ Mapping files

- Mapping files will pull data from the Azure database. It will map the GPS coordinates and the sidewalk to a CSV file using react-csv.
- Mapping files can also be created with a shapefile from ArcGIS which can then either be imported or hosted on NavigateLA.
- Depending on how much sidewalk data is collected mapping files can span from just 1 sidewalk segment to all sidewalk segments across Los Angeles

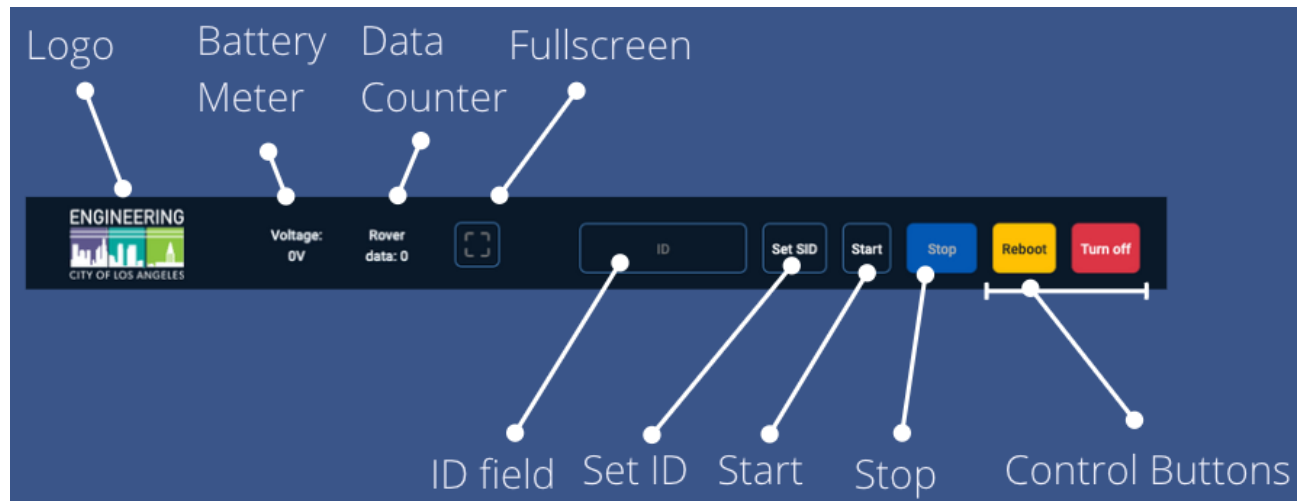
## 4.2 Task 2

### 4.2.1 Rover UI



*Rover UI*

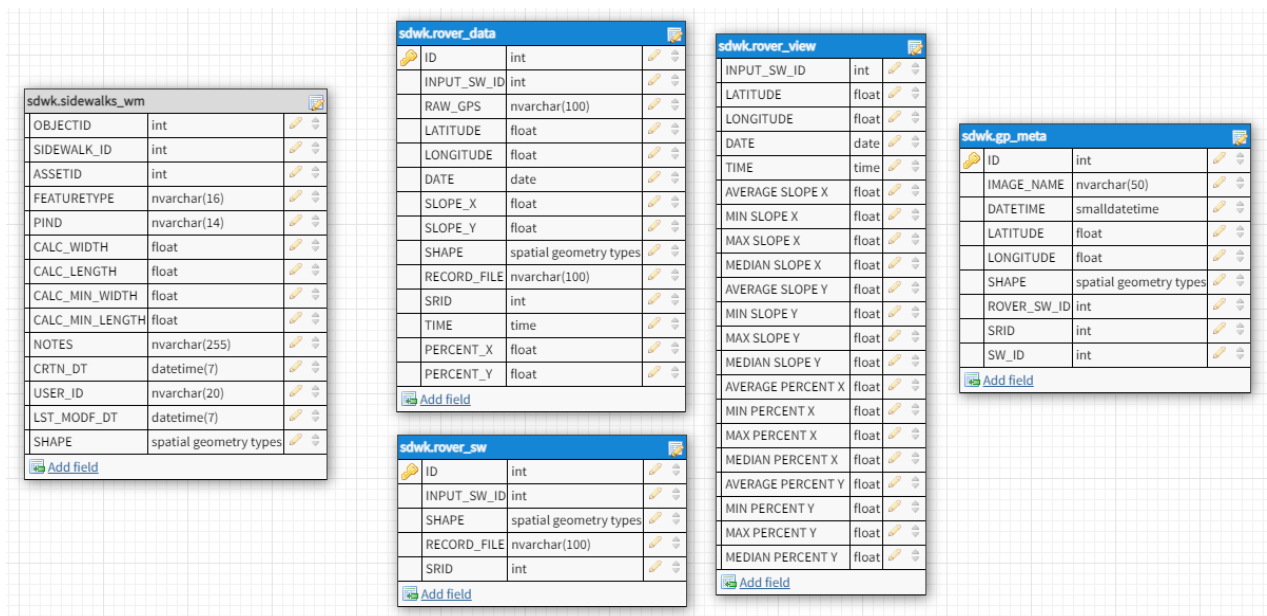
This Rover UI allows manual control of the Leo Rover. Input field for the Sidewalk ID corresponds to the sidewalk segment to start collecting data. Data is stored on the SD card on the rover to be post processed.



*Rover UI Navbar*

## 4.3 Task 3

### 4.3.1 Database Data Diagram





#### 4.3.2 Sidewalk Table

This dataset contains an inventory of City of Los Angeles Sidewalks and was digitized using a combination of G.I.S software, aerial imagery (2014 LARIAC), and geographic dataset of property/right-of way lines. Dataset is not actively being maintained. Last updated: 2/23/2021

**Table Name - sdwk.sidewalk\_wm**

Column name	Type	Descriptive name	Allow Nulls	Description
OBJECTID	INT	Object Field Asset Identifier	NO	A unique feature identifier populated by Los Angeles City Staff for internal use.
SIDEWALK_ID	INT	Sidewalk Identifier	NO	A unique feature identifier populated by Los Angeles City staff for internal use.
ASSETID	INT	Asset Identifier	NO	A unique feature identifier populated by Los Angeles City staff for internal use.
FEATURETYPE	NVARCHAR (16)	Feature Type	NO	Describes the type of feature class(sidewalk/ramp/driveway/etc) the data belongs to.
PIND	NVARCHAR (14)	Parcel Identification Number	NO	A unique Parcel Identification Number (PIN) for all parcels

				within the City of L.A. All Sidewalk related features will be split, non-overlapping, and have one associated PIN.
CALC_WIDTH	FLOAT	Calculated Width	NO	A generalized width of the feature calculated using spatial and mathematical algorithms on the feature. Widths are rounded to the nearest whole number. In cases where there is no value for the width, the applied algorithms were unable to calculate a reliable value.
CALC_LENGTH	FLOAT	Calculated Length	NO	A generalized length of the feature calculated using spatial and mathematical algorithms on the feature. Lengths are rounded to the nearest whole number. In cases where there is no

				value for the length, the applied algorithms were unable to calculate a reliable value.
CALC_MIN_WIDTH	FLOAT	Calculated Minimum Width	NO	In almost all cases where features have variable widths, the minimum width is used.
CALC_MIN_LENGTH	FLOAT	Calculated Minimum Length	NO	In almost all cases where features have variable widths, the minimum length is used.
NOTES	NVARCHAR (255)	Notes	YES	Short notes created by
CRTN_DT	DATETIME ( 7)	Creation Date	NO	Indicates date feature was created
USER_ID	NVARCHAR (20)	User Identifier	NO	Name of field agent responsible for feature input
LST_MODF_DT	DATETIME ( 7)	Last Modified Date	NO	Indicates date feature was revised/edited
SHAPE	GEOMETRY	Shape	NO	A polygon that represents paved pedestrian walkways. The polygon is constructed by

				several GPS (Latitude, Longitude) coordinates. The first coordinate of the polygon will also always be the last coordinate.
--	--	--	--	---

#### 4.3.3 Rover data table

The rover data table was created to store data extracted from rover output files. The table contains GPS coordinates, slope measurements, and information of the file the data originated from. All coordinates and geometry data are in Web Mercator(SR3857) format and can be used to locate a position on any map system using SR3857.

**Table Name - sdwk.rover\_data**

Column name	Type	Descriptive name	Allow Nulls	Description
ID	INT	Identifier	NO	Incremental number used for indexing.
INPUT_SW_ID	INT	Input Sidewalk Identifier	NO	Numeric identifier assigned by field agent. Identifiers should be referenced from the sidewalk table.
RAW_GPS	NVARCHAR(100)	Raw GPS Text	NO	Text string extracted directly from rover output csv file.
LATITUDE	FLOAT	Latitude	NO	Latitude coordinates extracted from

				GPS values in SR:3857 format.
LONGITUDE	FLOAT	Longitude	NO	Longitude coordinates extracted from GPS values in SR:3857 format.
DATE	DATE	Date	NO	Date indicating the recording of GPS data
SLOPE_X	FLOAT	Slope X	NO	X slope value extracted from rover data recording.
SLOPE_Y	FLOAT	Slope Y	NO	Y slope value extracted from rover data recording.
SHAPE	GEOMETRY	Shape	NO	Spatial point geometry created from latitude and longitude referencing GPS position rover data was recorded.
RECORD_FILE	NVARCHAR(100)	Recorded File Name	NO	File name of .csv file recorded data was extracted from.
SRID	INT	Spatial Reference Identifier	NO	Spatial reference identifier denoting the format GPS

				coordinates are in.
TIME	TIME	Time	NO	Time indicating the recording of GPS data
PERCENT_X	FLOAT	Percent Slope X	YES	X slope value in percentage
PERCENT_Y	FLOAT	Percent Slope Y	YES	Y slope value in percentage

#### 4.3.4 Rover sidewalk

The rover sidewalk was created to store spatial geometry generated per output file from the rover. Every GPS coordinate collected from the output file is used to create a polygon (SHAPE) that can be used to generate an outline of the portion of sidewalk traversed by the rover. The default spatial reference id is 3857 and all relative coordinate data is formatted in that manner.

**Table Name - sdwk.rover sw**

Column name	Type	Descriptive name	Allow Nulls	Description
ID	INT	Identifier	NO	Auto incrementing number used for indexing.
INPUT_SW_ID	INT	Input Sidewalk Identifier	NO	Numeric identifier assigned by field agent. Identifiers should be referenced from the sidewalk table.

SHAPE	GEOMETRY	Shape	NO	A polygon that represents the sidewalk positions the rover traversed. The polygon is constructed by several GPS (Latitude, Longitude) coordinates. The first coordinate of the polygon will also always be the last coordinate.
RECORD_FILE	NVARCHAR(100)	Recorded File Name	NO	File name of .csv file recorded data was extracted from.
SRID	INT	Spatial Reference Identifier	NO	Spatial reference identifier denoting the format GPS coordinates are in.

#### 4.3.5 Sidewalk GoPro Metadata

The Sidewalk GoPro Metadata table was created to store metadata extracted from GoPro images. The GPS coordinates from the metadata is useful to associate the image taken to a position on the map. The IMAGE\_NAME can also be used to locate the image file from the Azure Blob store. The ROVER\_SW\_ID references the INPUT\_SW\_ID from the sdwk.rover\_sw table. To find the sidewalk id associated with the image, a geometric point is created from the latitude and longitude of the metadata to find the nearest geometry from the sdwk.rover\_sw table to that point and assign the associating INPUT\_SW\_ID.

**Table Name - sdwk.gp\_meta**

Column name	Type	Descriptive name	Allow Nulls	Description
ID	INT	Identifier	NO	Auto incrementing number used for indexing.
IMAGE_NAME	NVARCHAR(50)	Image File Name	NO	Image file name where metadata was extracted from. This is used to locate the image from the Azure blob store.
DATETIME	SMALLDATETIME	Date time	NO	Date and time of image recording.



LATITUDE	FLOAT	Latitude	NO	Latitude position for where the image was taken. Coordinates are in SR:3857 format.
LONGITUDE	FLOAT	Longitude	NO	Longitude position for where the image was taken. Coordinates are in SR:3857 format.
SHAPE	GEOMETRY	Shape	NO	Spatial point geometry created from latitude and longitude referencing GPS position GoPro data was recorded.

ROVER_SW_ID	INT	Rover Sidewalk Identifier	YES	The sidewalk id the image was estimated to be recorded at. A geometric point is created from the longitude and latitude values and used to find the closest rings to that point and the sidewalk id is taken from the row containing the rings.
SRID	INT	Spatial Reference Identifier	NO	Spatial reference identifier denoting the format of GPS coordinates
SW_ID	INT	Sidewalk ID	YES	Corresponding closest Sidewalk ID to the image GPS coordinate

#### 4.3.6 Sidewalk Rover Data View

The Sidewalk Rover View was created to organize collected rover data due to clustering of GPS coordinates. The contained aggregate data is useful in determining the sidewalk quality at a given location. The grouping by unique coordinate is used to organize all of the rover slope data.

##### View Name - sdwk.rover view

Column name	Type	Descriptive name	Allow Nulls	Description
INPUT_SW_ID	INT	Input Sidewalk Identifier	NO	Numeric identifier assigned by field agent. Identifiers should be

				referenced from the sidewalk table.
LATITUDE	FLOAT	Latitude	NO	Latitude coordinates extracted from GPS values in SR:3857 format.
LONGITUDE	FLOAT	Longitude	NO	Longitude coordinates extracted from GPS values in SR:3857 format.
DATE	DATE	Date	NO	Date indicating the recording of GPS data
TIME	TIME	Time	NO	Time indicating the recording of GPS data
AVERAGE SLOPE X	FLOAT	Average Slope X	NO	Average Slope X value for a GPS coordinate
MIN SLOPE X	FLOAT	Min Slope X	NO	Min Slope X value for a GPS coordinate
MAX SLOPE X	FLOAT	Max Slope X	NO	Max Slope X value for a GPS coordinate
MEDIAN SLOPE X	FLOAT	Median Slope X	NO	Median Slope X value for a GPS coordinate
AVERAGE SLOPE Y	FLOAT	Average Slope Y	NO	Average Slope Y value for a GPS coordinate
MIN SLOPE Y	FLOAT	Min Slope Y	NO	Min Slope Y value for a GPS coordinate
MAX SLOPE Y	FLOAT	Max Slope Y	NO	Max Slope Y value for a GPS coordinate

MEDIAN SLOPE Y	FLOAT	Median Slope Y	NO	Median Slope Y value for a GPS coordinate
AVERAGE PERCENT X	FLOAT	Average Percent X	NO	Average Percent X value for a GPS coordinate
MIN PERCENT X	FLOAT	Min Percent X	NO	Min Percent X value for a GPS coordinate
MAX PERCENT X	FLOAT	Max Percent X	NO	Max Percent X value for a GPS coordinate
MEDIAN PERCENT X	FLOAT	Median Percent X	NO	Median Percent X value for a GPS coordinate
AVERAGE PERCENT Y	FLOAT	Average Percent Y	NO	Average Percent Y value for a GPS coordinate
MIN PERCENT Y	FLOAT	Min Percent Y	NO	Min Percent Y value for a GPS coordinate
MAX PERCENT Y	FLOAT	Max Percent Y	NO	Max Percent Y value for a GPS coordinate
MEDIAN PERCENT Y	FLOAT	Median Percent Y	NO	Median Percent Y value for a GPS coordinate

## Task 3

### Web API Server

- Starting the server only needs NodeJS to run the main index file of the code directory. Errors thrown will cause the server to terminate and it is advised to use utility tools to restart the server automatically when errors are encountered.
- When an endpoint is sent a request, the route handling within the primary index file delegates all actions to the proper module designated for the route. The route handler is responsible for providing the module the proper data needed by the modules and the instance of the database client.
- The database client within the API server is only instantiated once in the main index file and passed as an argument to modules from the route handlers. The database client configuration must be set up prior to its instantiation.
- The primary endpoint for data requests operates in three stages. The first is the processing of the query parameters from the request body. All query parameters are received as strings regardless of the intention of the requester and the processing step is to parse all numeric strings to numbers. The second stage constructs a SQL statement from the JSON object using specified keys, operator keys list, and arrays or nested objects. The final result from stage 2 is a SQL statement where in the final stage it is executed on the database through the database client and the results are returned back through the response handler.
- The endpoint for finding the closest sidewalk for a given coordinate utilizes only two parameters, one contains a pair of coordinates and the other the SRID of the coordinate. The SRID is used to determine whether the coordinate requires conversion to SRID 3857, which is the spatial reference system used by the sidewalk table. The coordinates are composed in a SQL statement that searches for the SIDEWALK\_ID closest to the given coordinate. Depending on the server load, this process can take around 5-7 seconds per coordinate.
- The coordinate conversion endpoint does not require the database client and although it is normally a read only type request, the endpoint itself is a POST type request. Due to the character limits of URL/URI and a list of coordinates can number largely, the request type must be a POST type to allow the use of the request body. The route simply passes the request body, an array of coordinates, and the designated module will iterate each pair of coordinates and convert them appropriately specified by the URI query parameters.
- The CSV upload route is the only route that spawns child processes. The route handler extracts the target table and origin source(rover or GoPro), and assigns the proper Python script to process and upload the CSV file. The CSV files are received through the request body, parsed into CSV from the request body, written to a temporary directory, where it can be read by the Python script and deleted once processed and inserted into the database. All file and data processing is handled through Python. The route also ensures unprocessed files within the temporarily created directories are to be removed after a set time period to prevent unnecessary storage waste.

## 5.0 Policies and Tactics

### 5.1 Choice of which specific products used

- Task 1
  - Web application
    - React
      - React uses Node.js for its backend and HTML in the form of JSX.
    - HTML, CSS, and Bootstrap
    - Google Map Javascript API
    - GoPro Fusion Studio
    - Python
      - Manually extracts all GoPro image metadata
  - Mapping files
    - React-csv library
      - This library was chosen for its features that make reading, formatting, and writing data into a CSV straight forward. React-csv also lets the browser handle where downloads will go.
    - ArcGIS will need to be used for FALL 2022 color coding.
      - Program available to automate mapping file creation. Forums and documentation thoroughly explains how the application may be used and provides sample use cases.
      - Utilizes Python code for its command prompt which will handle automation
- Task 2
  - Rover UI
    - HTML
      - Used as the standardized system for tagging text files to achieve font, color, graphic, and page structure.
    - JavaScript
      - Object oriented programming language used to create interactive effects within the Rover UI.
    - CSS
      - Style sheet language used for describing the presentation of the Rover UI.
    - roslibJS
      - Core Javascript library for interaction with ROS from the browser.
    - Leo Rover was chosen as a replacement for our previous rover “Robecca” because it's more versatile, price friendly, and is a good platform to base a fleet of rovers on.

- GoPro Fusion was chosen as our camera because it offers a 360-degree view of the rovers surrounding with only one product.
- Task 3
  - Database
    - Azure SQL database was chosen by our corresponding liaisons because it's what they are currently using themselves.
    - Azure Blob storage was chosen as a solution for image storage because it integrates to Azure DB nicely and it's the most cost-effective method available to the BOE

## 5.2 Plans for ensuring requirements traceability

- Task 1
  - Web application
    - Backend is actively extracting data from the Azure storage and SQL Server database. It is not storing or using local data.
  - Mapping files
    - Automation scripts create shapefiles that meet the design requirements. Design requirements include color schemes based on BOE's prioritization and scoring system, annotations for slope, etc

## 5.3 Plans for testing the software

- Task 1
  - Web application
    - Users can view an image. The administrators can confirm that the image that is shown and its corresponding data are related.
    - Users can click on a map coordinate and the data will change accordingly.
    - React local server will be created for testing purposes
  - Mapping files
    - When running the web application we developed, it pulls data from the Azure database and creates a CSV file.
    - The shapefiles must display the shape of a sidewalk with annotations and any needed color schemes which is to be developed in FALL 2022. (applying color schemes based on BOE's prioritization and scoring system which is outlined in the BOE report under Damage Severity Matrix).
- Quality Control

## 6.0 Detail System Design

### 6.1 Raspberry Pi 3

#### 6.1.1 Responsibilities

The role of this module will be to compute and store data captured by the rover modules such as Camera and Accelerometer.

#### 6.1.2 Constraints

Possible constraint of this module would be the memory size of the MicroSd Card attached to the rover.

#### 6.1.3 Composition

A description of the use and meaning of the subcomponents that are a part of this component.

Uses/Interactions

User interaction to power the Raspberry Pi 3

#### 6.1.4 Resources

This is the main module that will power the accelerometer and camera.

They are dependent on Raspberry Pi 3

#### 6.1.5 Interface/Exports

CSV file will export data from the MicroSd card into the database.

Interface Linux.

### 6.2 Rover User Interface

#### 6.2.1 Responsibilities

The primary responsibility of the rover interface is to allow the user to access, move, and to automate data collection using the rover.

#### 6.2.2 Constraints

Constraints of this component would be based on the sidewalk condition, any debris would give inaccurate results.

#### 6.2.3 Composition

Extra camera added on the rover is a GoPro. It is responsible for capturing photos of the sidewalk where data was recorded.

#### 6.2.4 Uses/Interactions

The rover user interface will be used to control the rover movements and ability to collect data. The rover user interface will interact with the Bureau Of Engineering field workers.

#### 6.2.5 Resources

Rover UI



## 6.3 NavigateLA

### 6.3.1 Responsibilities

From the images taken from the GoPro camera attached to the Rover. The functionalities include measuring the XY (vertical and horizontal) displacements on the sidewalk and eventually categorizing the images according to their condition.

### 6.3.2 Constraints

Measuring the vertical displacement is close to impossible. The angle of the images being taken from the camera doesn't show the distance from an overhead view.

### 6.3.3 Composition

We use a Python library that is compatible with processing functionality. We currently use Pycharm IDE to utilize those components.

### 6.3.4 Uses/Interactions

NavLA web segment implemented will generate the sidewalk data packaging dates, when the measurements were made, categorizing into dates.

### 6.3.5 Resources

Beginning stages don't require any resources other than the images used to learn and test with algorithms.

### 6.3.6 Interface/Exports

Used Python library built in for analyzing collected data.

## 6.4 Web application and mapping files

### 6.4.1 Responsibilities

The web application is an interface for users to view a collection of rover data. The mapping files script will create CSV files automatically that can either be imported or hosted on NavigateLA.

### 6.4.2 Constraints

Web applications will be developed in a test environment locally. If the liaison provides a web server, the web application will be moved to a production environment. The mapping files will be imported to NavigateLA. If many mapping files are created using sidewalk data, the mapping files can be hosted on NavigateLA. This depends on the amount of sidewalk data collected, GoPro images available, and a database to host the mapping files.

#### **6.4.3 Composition**

The web application is the web console that users interact with. The mapping files are automation scripts that will create a CSV file for visualization

#### **6.4.4 Uses/Interactions**

The web application and the mapping files scripts use the image processing scripts and the Azure database.

#### **6.4.5 Resources**

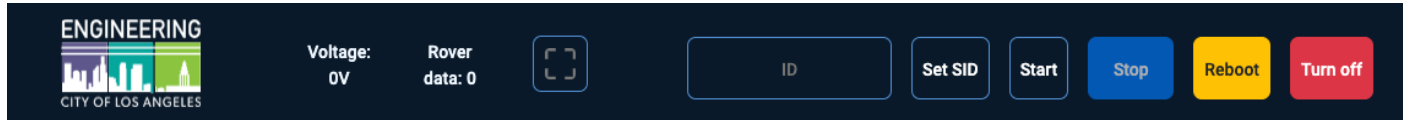
Azure database

#### **6.4.6 Interface/Exports**

Web application is the interface and the mapping files scripts create CSV files.

## 7.0 Rover UI Design

### 7.1 Rover UI Navbar



Located at the uppermost section of the Rover UI:

- BOE logo.
- *Battery meter* that displays the rover battery.
- *A data counter* that represents the number of data segments collected.
- *Fullscreen button* that will transform the view to full screen of the device being used.
- *An ID field* that accepts the ID of the section to be collected.
- *SetID button* which sets the ID from the section ID field.
- *Start button* to begin collecting data.
- *Stop button* that ends the data collection.
- *Reboot button* that reboots the rover system.
- *Turn off button* that shuts down the rover.

### 7.2 Rover UI Direction and Stop Button



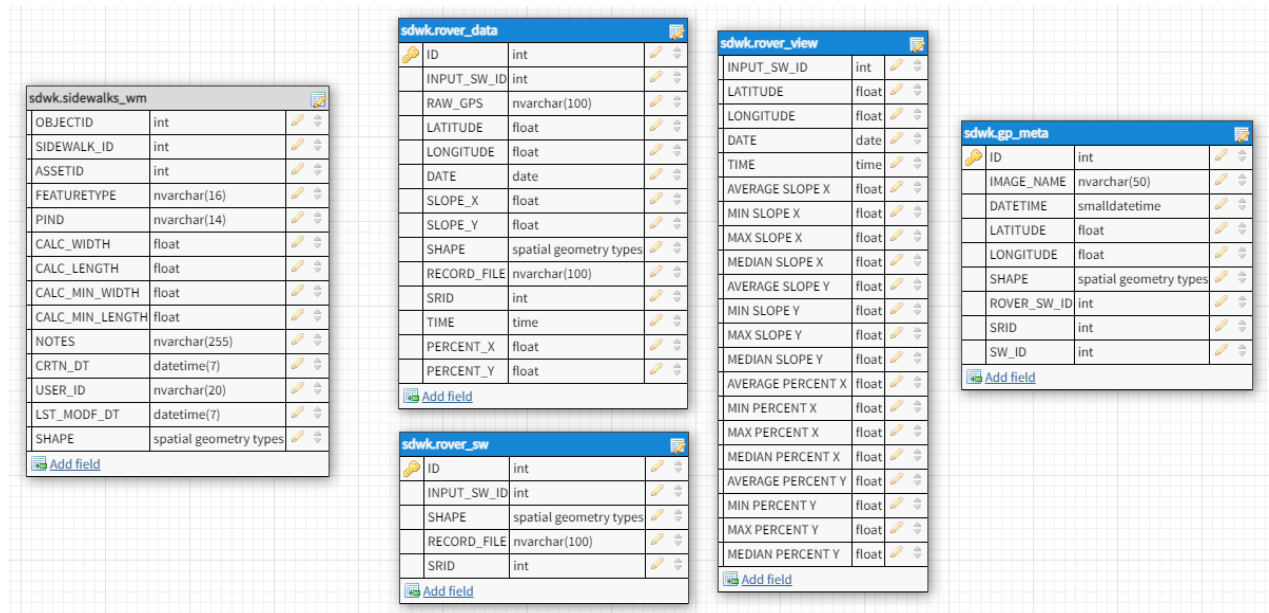
Located on the lower half of the Rover UI:

- Transfers movement of the directional button controls to the direction the rover will navigate towards.
- The stop button stops all rover movement.

## 8.0 Database Design

The database is utilized by all aspects of the project, linking the existing data from NavLA to new data collected by the rover--providing access web server which displays the data in real-time. In addition, it provides access and stores the images from the GoPro which can then be directly implemented with the image processing side of things.

### 8.1 Relational Schema



### 8.2 Diagram Description

The relational schema as seen in the diagram is organized by the various sources of data in mind.

The leftmost part is the existing data provided by BOE:SWBot, and property tables. It includes PIN IDs and asset sidewalk IDs which are essential in pinpointing specific properties and sidewalk locations. Navigate LA data will serve as geographical reference points for the GoPro data and the rover data listed.

The rightmost data, containing the two Rover Data Tables and GoPro Image Data Table, would be extracted from the rover itself after collection and uploaded in their respective tables. The rover\_data table consists of collected GPS and slope data along with other identifiers such as input sidewalk ID and the spatial reference ID. In order to represent the collected data as a shape, the data is processed and stored in the rover\_sw table. The GoPro metadata is contained in the gp\_meta table, which holds the GPS data extracted from the taken images as well as useful identifiers such as the sidewalk ID.

### 8.3 Azure blob Storage implementation

Azure Blob Storage is required in order to store a large amount of JPGs--such as the front, rear,

and rendered Images. Image IDs from the GoPro picture data will serve as a link for the images stored in the blob storage. This method allows us to create a much more efficient method of storing images and sharing them across different platforms, such as our UI and web applications.

## 8.4 Data Collection

Data sources include:

1. Rover
  - a. The main focus as it holds key information that will be used by BOE to prioritize the locations in need of repairs.
  - b. GPS Coordinates (Precise Lat and Longitude)
  - c. Timestamps
  - d. Leveler - Vertical and Horizontal displacement of the sidewalk
2. GoPro Fusion
  - a. Forward and Rear facing camera to create a 360 Degree image on the rovers surroundings
  - b. Rendered Images
  - c. Python code to exfiltrate the EXIF/METADATA from each image which provides another DB table to populate
3. Existing data - Mapped by the BOE on NavLA

## 8.5 Future Implementation

Future implementations consist of things such as

1. Data Manipulation
  - The way data is correlated, either by location or specific timestamps
  - Automation of data uploads
2. Data Visualization
  - Azure offers Tools which help visualize our data, which offers a unique perspective that could be implemented into our applications.
3. Image processing
  - Azure AI will be utilized to introduce severity levels into our dataset and help reduce the manual labor and reach our goal faster and more productively.

## 9.0 User Interface

The user interface tasks are Task 1 for the web application and Task 3 for the rover UI.

### 9.1 Overview of User Interface

#### 9.1.1 Task 1, Web Application

The web interface will receive image data from Leo Rover's camera to be transferred to the website. BOE uses an Azure database for their backend, which we will integrate into our web application.

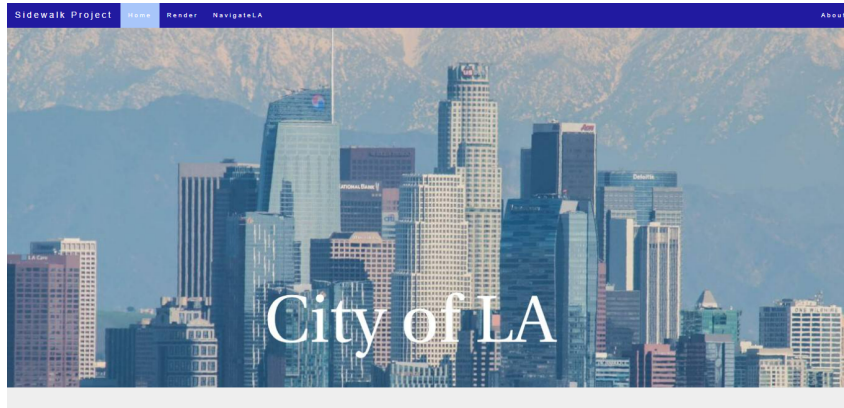
Our web application will include a page to display our data. The data to be displayed includes from the rover, longitude and longitude slope data, Global Positioning System (GPS) data as well as the image name and date when what image was taken. The images that will be displayed come from the GoPro fusion camera on the rover, and we will also be displaying the GoPro metadata which includes, longitude and latitude data. The web application will have the ability to iterate through the image with a previous button and next button. The user will be able to pan the image and zoom in and out of the image.

The web application will have additional pages which will describe the purpose of our project, the overall description, the environment where our project was worked on, and the algorithms used in this project.

### 9.2 Screen Frameworks or Images

#### 9.2.1 Task 1, Web Application

The following are wireframes and actual frontend images:



## Sidewalk Project



The City of Los Angeles, Bureau of Engineering contains over 11,000 miles of sidewalks. When a segment of sidewalk does not settle evenly or has been raised up by tree-root growth, the sidewalk becomes uneven. This can create pedestrian hazards. In addition, the City is obligated to ensure that its sidewalks conform to Federal ADA standards, which limit the extent to which a sidewalk may slope.

## Leo Rover



### Leo Rover

Leo Rover is an outdoor Robotics Development Kit. It is open-source and built on RaspberryPi. With video streaming and driving UI ready out-of-the-box.

[Learn More](#)

ENGINEERING  
THE CITY OF LOS ANGELES

5151 State University Dr,  
Los Angeles, CA 90032

#### CONTACT US

Phone: 1-800-999-9999  
Email: [random@gmail.com](mailto:random@gmail.com)

#### HELPFUL LINK

[Terms of Use](#)  
[Privacy Policy](#)  
[FAQ](#)

Copyright © 2022 All Rights Reserved by Random Company.

Sidewalk Data

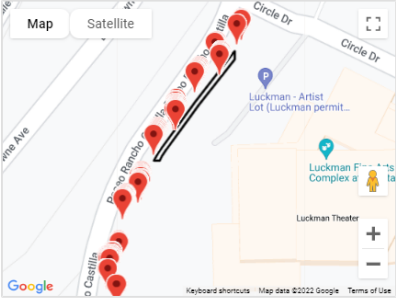
Section ID  
998279

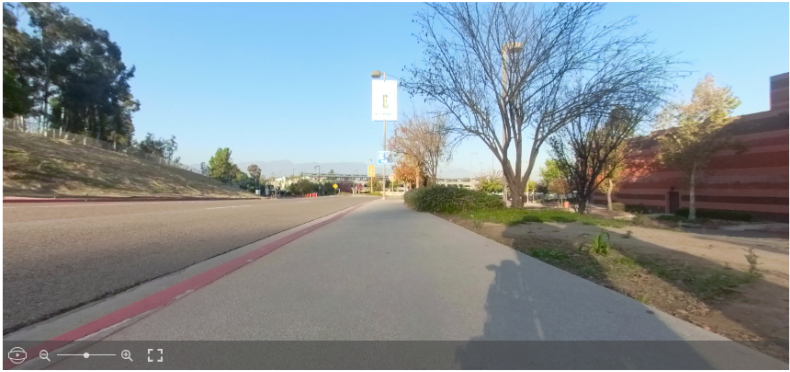
PHOTO\_1913.jpg  
Date: 2021-11-10

SLOPE X ↔  
0°

SLOPE Y ↓  
0°

Global Positioning System (GPS)  
Latitude: 34.069226999999415  
Longitude: -118.16949633333134





← Prev

Next →





## Why is this project being Developed?

The City of Los Angeles, Bureau of Engineering maintains over 11,000 miles of sidewalks. When a segment of sidewalk does not settle evenly or has been raised up by tree-root growth, the sidewalk becomes uneven. This can create pedestrian hazards. In addition, the City is obligated to ensure that its sidewalks conform to Federal ADA standards, which limit the extent to which a sidewalk may slope.

### Sidewalk Obligations And Liabilities

♦ Liability between municipalities and landowners for the condition of the sidewalk and for injuries sustained by those using the sidewalk due to defective sidewalk conditions is the subject of lawsuits and statutory provisions. In California, municipalities and counties usually own the sidewalks next to private property, but California state law long enacted states that the landowners are responsible for maintaining the sidewalk fronting their property in a safe and usable manner. ♦

[Learn More](#)



## 2021-2022 Student Goals

This is the fifth term of a multi-year project. In the last term, the rover was made capable of moving autonomously, measuring crossing slopes and running slopes, collecting GPS data, and take photo images.

In this term, we will focus on developing various software that:

- 1) Renders the photo images with slope data and GPS data
- 2) Processes images such as object segmenting and texture processing
- 3) Assists field crew while they are collecting data in the field (i.e. a user-friendly mobile app/web app).
- 4) In addition, develop a database schema and backend server to store and manage raw data.

## Softwares & Equipment



Rover



Azure



Node.js



Javascript



HTML/CSS



React



Image Segmentation



Google Maps API



5151 State University Dr,  
Los Angeles, CA 90032

CONTACT US  
Phone: 1-(800)-999-9999  
Email: random@gmail.com

HELPFUL LINK  
Terms of Use  
Privacy Policy  
FAQ

Copyright ♦ 2017 All Rights Reserved by Random Company.

Sidewalk ProjectHomeRenderNavigateLAAbout

2021 ▾

November 10 2021

Package

2022 ▾

February 01 2022

Package

February 24 2022

Package

February 25 2022

Package

March 01 2022

Package

March 03 2022

Package

ENGINEERING  
CITY OF LOS ANGELES

5151 State University Dr,  
Los Angeles, CA 90032

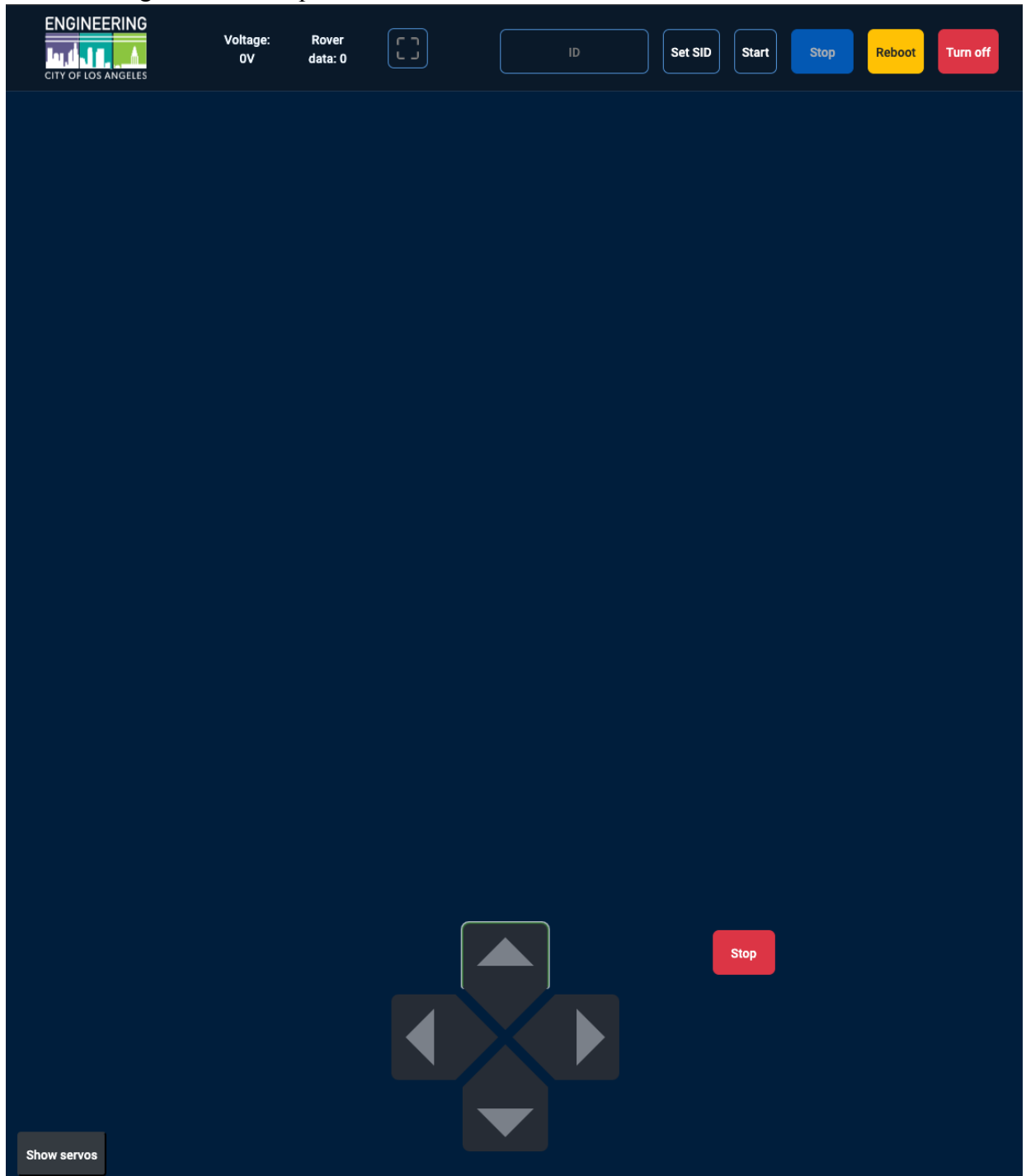
CONTACT US  
Phone: 1-(800)-999-9999  
Email: random@gmail.com

HELPFUL LINK  
Terms of Use  
Privacy Policy  
FAQ

Copyright ♦ 2022 All Rights Reserved by Random Company.

### 9.2.2 Task 3, Rover UI

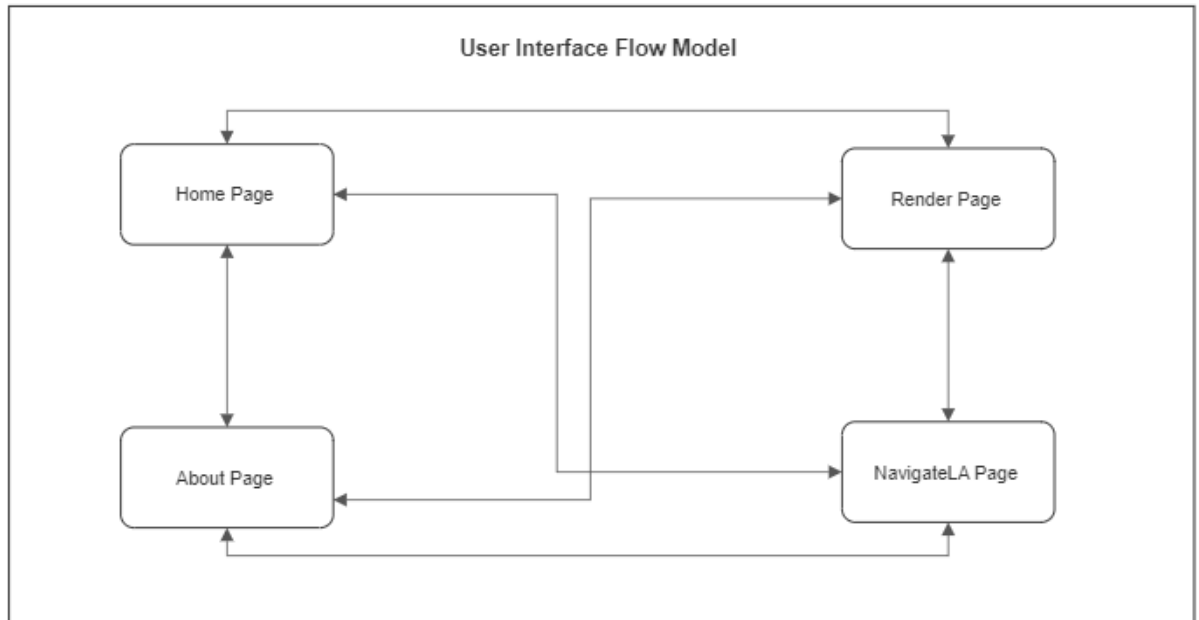
The following is a screen capture of the Rover UI that the user will use.



## 9.3 User Interface Flow Model

### 9.3.1 Task 1, Web Application

The Home, Render, NavigateLA, and About page can freely navigate between each other. As for the NavigateLA tab, a user is able to download any of the packages provided in the tab.



## 10.0 Requirements Validation and Verification

Requirement	Component	Test
	Module	
Raspberry Pi 3 shall collect data from modules and store the data.	Raspberry Pi 3	To be determined in Spring
Accelerometer shall measure the grade of sidewalk and send the raw data to Raspberry Pi 3.	Accelerometer	To be determined in Spring
Camera shall be responsible for taking pictures.	Camera	To be determined in Spring

Real-Time Kinematic GPS unit to record centimeter accurate coordinates.	GNSS Device	To be determined in Spring
---	-------------	----------------------------

## 11.1 Glossary

For the purposes of this project, the following terms are defined as follows:

BOE	City of Los Angeles, Bureau of Engineering
SDD	Software Design Document
SRS	Software Requirements Specification
Rover	Device with wheels that will display the GUI software.
IMU	Inertial Measurement Unit
DB	Database
NavLA	NavigateLA (Site)
EXIF	Exchangeable Image File
IDE	Integrated Development Environment
GUI	Graphical User Interface
Task 1	Web application and mapping files
Task 2	Rover UI
Task 3	Database

## 12.0 References

- NavigateLA, <https://navigatela.lacity.org/navigatela/>
- Sidewalk Repair Program Prioritization and Scoring System Council File 14-0163-S3, from the City of Los Angeles, Bureau of Engineering
- Azure SQL database - <https://docs.microsoft.com/en-us/azure/azure-sql/>
- Azure Blob - <https://docs.microsoft.com/en-us/azure/storage/blobs/>
- Leo Rover - <https://www.leorover.tech/>
- GoPro Fusion - [https://gopro.com/content/dam/help/fusion/manuals/Fusion\\_UM\\_ENG\\_REVC.pdf](https://gopro.com/content/dam/help/fusion/manuals/Fusion_UM_ENG_REVC.pdf)