Software Design Document
for
Sidewalk Slope Monitoring System

Version 2

Prepared by Jan Bautista, Hua Chen, Abigail Garcia, Ana Guardado, Cristina Munteanu, Pabasara Navaratne, Alexis Pena, Beatriz Ruiz, Aoqian Wang

Department of Public Works and Bureau of Engineering, LA City

CSULA Senior Design 2020-2021 / LA City Bureau of Engineering

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Abigail G. + Alex P.	12/8/2020	Fall 2020 Draft	1
Pabasara N.	12/9/2020	Added basic descriptions to sections 1 and 2	1
Ana G.	12/10/2020	Fixed formatting for headers, changed table of contents for automatic updates. Updated descriptions, contents, explanations for Section 1, *Introduction.* Changes pending approval of SDD lead, Pabasara	.2
Pabasara N. + Jan B.	12/10/20	Added to section 2.3,2.4, 4, 5.1, 6.3, and update the table of contents. Fixed headers.	.2
Cristina M. + Beatriz R.	12/10/20	Fall 2020 Draft	1
Ana G.	12/11/2020	Added to sections 2, 3, 4, 5, 6 from a task 1 perspective.	.2
Alexis and Abigail	5/12/2021	Update Information on Database	.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. The template follows the format:

- Main sections are identified by size 20, bold Times New Roman font
- Sub-sections are identified by size 14, bold Times New Roman font

1.3 Intended Audience and Reading Suggestions

This document is intended to aid the team members listed in 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 breakdown 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 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 2021.

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 sidewalk, 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. Once the images are collected, scripts produced during previous years' efforts will be used to parse out Exchangeable Image File (EXIF) data from the GoPro images. 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 to as Task 2.
3. A web application will display the collected images, one at a time, with its related EXIF and image processing data. 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 to Task 1.
4. A database developed by the team will contain the parsed EXIF data and the image processing scripts' output. The database is hereby referenced to as Task 4.

2.0 Design Considerations

2.1 Assumptions and Dependencies

- Rover expectations
    - Cracks and holes on the sidewalk shall be minimal.
    - Battery 4hrs of nominal driving or 8hrs 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.
        - Web application will be used to view damage and assign numerical scores to sidewalk segments.
        - Web application will be able to tell where sidewalk images were taken from based on coordinates.
        - Web application will be able to navigate between sidewalk images.
    - 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 hosted on NavigateLA, if and only if, mapping files are kept on a database managed by BOE.
- 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 always keep an eye on the rover.
        - User will manually extract data from rover's memory cards and save as CSV files.
    - Database
        - User will manually input data into Azure DB -- with an exception to some automation in the process.
        - User will manually input location data fields after every use.
        - User will manually link rover data to Navigate LA data

-
    -
        - User will manually input location data fields after every use.

2.2 General Constraints

- Task 1
    - Web application
        - Web application can use 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 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.
        - 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 Python.
        - Web application frontend will be done using HTML/CSS and Bootstrap.
        - Web application mapping will be dones using Google Maps API, similar to BOE.
    - Mapping files
        - 3rd party application must have a command language to develop automation scripts.
        - Mapping files must be generated by scripts using the 3rd party application's command language.
        - 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.

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 completed.
    - 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 the 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 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.
    - Meeting with the advisor and BOE will be required to determine how users can best view mapping files.
- Task 2
  - Image Processing
    - Implement image segmentation and texture processing methods on images to easily help find the x and y displacement of a sidewalk
    - Count pixels between two points.
    - Calculate x and y displacement between two 3D points.
- Task 3
  - Image reading structure for the image processing.
  - Utilize sidewalk measurements
  - Conjoin the four tasks
  - Redesign Rover Ui control buttons for easier recognition
  - Test Ui functionality on Rover
- Task 4
  - Rover
    - Build Leo Rover
    - Implement last year's project onto the Leo Rover.
    - Implement this year's tasks on the Leo Rover
  - Database
    - Design a database schema which will include all aspects of the project. Including but not limited it too, rover, UI's, and NavLA.
    - Create a way to bridge rover data with NavLa data. This might require the team to map sidewalk coordinates into their own table and use that to correlate with the existing tables. An algorithm will then be needed to loop through all the coordinates and make that correlation.
    - Create the necessary SQL statements to create said database in an Azure environment provided by the BOE to begin storing important data.

        - Implement Azures Blob storage within the BOE's environment to be able to store JPG's and CSV files.
        - Implement data upload automation
        - 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, HTML/CSS and python, from web development courses to build the frontend and backend of the site.
        - Refer to Mozilla and Django's documentation for best practices and setting up the environment.
        - Will use Google API as a reference to iterate through images
    - Mapping files
        - Leverage existing automation efforts to create DWG files
        - Reference AutoCAD's forums to explore other user's methods and processes when developing scripts using AutoCAD's command language
        - Run through manual steps in developing DWG files, to narrow down what is required for the automation scripts
- Task 2
    - Use the Grabcut Algorithm from the OpenCV library to easily remove background noise and segment sidewalks from captured Lidar images.
    - Use Euclidean distance method when calculating the distance between two points to find x and y displacement in pixels.
    - Use the Euclidean distance method to find the displacement between two 3D points in meters then convert to centimeters.
- Task 3
    - Rover User Interface allows field users to control the rover and collect sidewalk data.
- Task 4
    - Database allows for access to storage from all aspects of the project.
    - Sources of data are the rover with numerical data as well as the GoPro with image and EXIF data collected upon testing; incorporated with the corresponding sidewalk data provided by BOE.
    - Organized in Azure Data Studio to efficiently manipulate and connect data from all sources including the rover as well as other image data.
    - With the substantial number of images involved, Azure Blob storage is implemented for easy access from other tasks.

3.0 Architectural Strategies

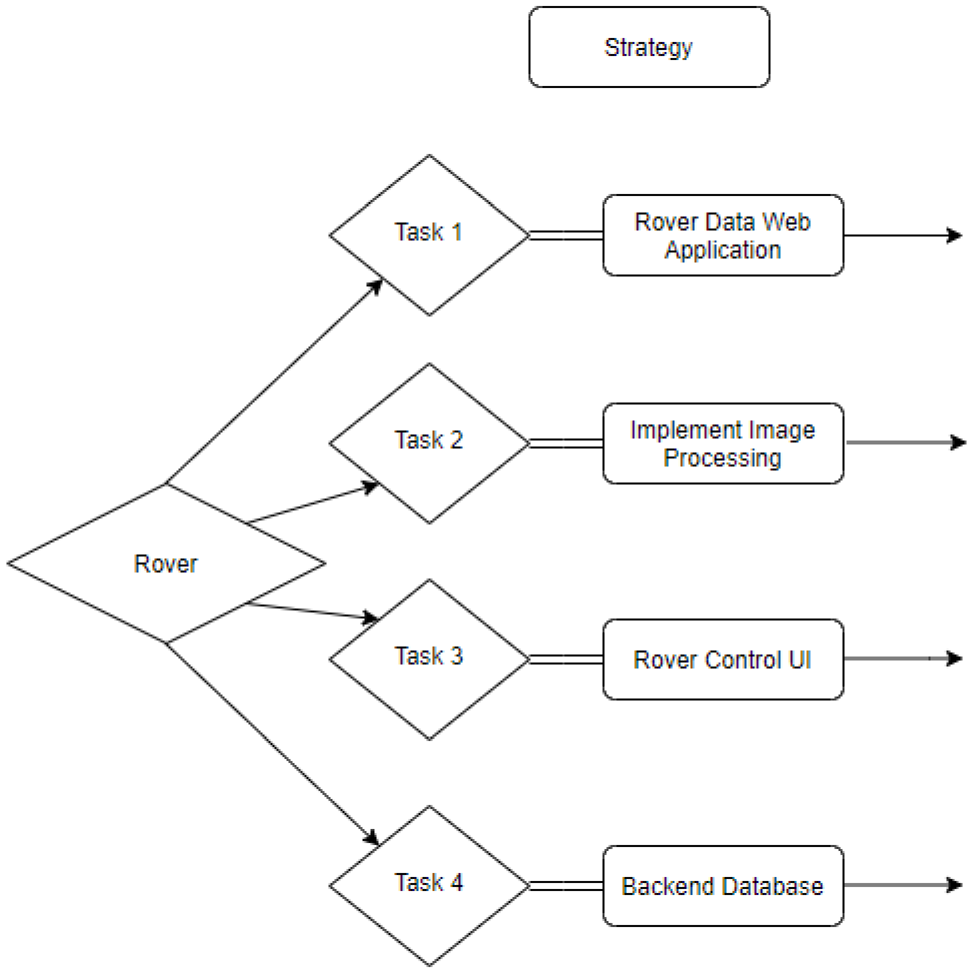
3.1 Task 1

- Web application
  - Used Django to use HTML, CSS, Bootstrap for frontend and Python for backend.
    - Python libraries are heavily used in parsing and processing image data both in task1 and in other tasks.
    - Enables teams to utilize each other's scripts and combine efforts without being concerned with language compatibility.
    - Django also uses templates that make it easier in developing the backend.
    - Django testing environment can easily be created. Existing environment can easily be replicated following Django and Mozilla's documentation.
    - Django, Mozilla, and Python have an extensive set of documentation that we have heavily depended on to ensure our product is created with best practices.
  - Will use Google API as a reference to iterate through sidewalk images
  - Utilizing existing scripts to parse EXIF data from GoPro images
    - Scripts were created during previous years' efforts. Scripts will be reused.
  - Depending on availability of web server from liaison, the Django environment will be recreated for a production environment. Dependencies on liaison's web server application may also cause us to change our use from Django to another widely supported web server.
- Mapping files
  - Used AutoCAD to automate the creation of DWG files that will be mapped on NavigateLA.
    - AutoCAD's forums have many posts, notices, explanations, and pages regarding automation and how to use AutoCAD's command language.
    - AutoCAD is widely used and available to students for free from the university.
    - AutoCAD supports location and GPS coordinates, which allows us to use EXIF GPS data and map it easily on NavigateLA.
    - AutoCAD DWG files are supported by NavigateLA.
  - Other mapping applications such as ArcGIS are available and their file types are supported by NavigateLA, but ultimately AutoCAD was the best option.
    - 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.
    - AutoCAD 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.
    - Users would be able to select a layer 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 is forgotten.
    - However, in order for the layer to be hosted on NavigateLA, large amounts of sidewalk data and fully functioning automation scripts are required. This would develop many, or perhaps a large DWG file, that covers a large portion of the city of Los Angeles. We can then request a database from the liaison to host our data so that NavigateLA users can select the layer.

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, implement image processing, 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. Users can switch back and forth between images, use the auto feature so that the position in the sidewalk moves forward showing many images, or search for specific images based on GPS coordinates and image name.
  - 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 database. The console contains data from the image processing team, too.
- Mapping files

- Mapping files will pull data from the Azure database. It will map the GPS coordinates and the sidewalk to a DWG file in AutoCAD using automation scripts leveraging AutoCAD's command language.
- The DWG file can then either be imported or hosted on NavigateLA.
- The DWG file can provide an overview of the entire city of Los Angeles. Users can also zoom in to a section of the map and view annotations or additional notes created during the automation scripts.

4.2 Task 2

Implements the functionalities to process the images rendered and stored by the rover camera (lidar). Currently, the program reads an image and applies image processing algorithms using the OpenCV library to isolate the sidewalk and measure data..

Algorithms in use:

Image Thresholding and Canny Edge Detection

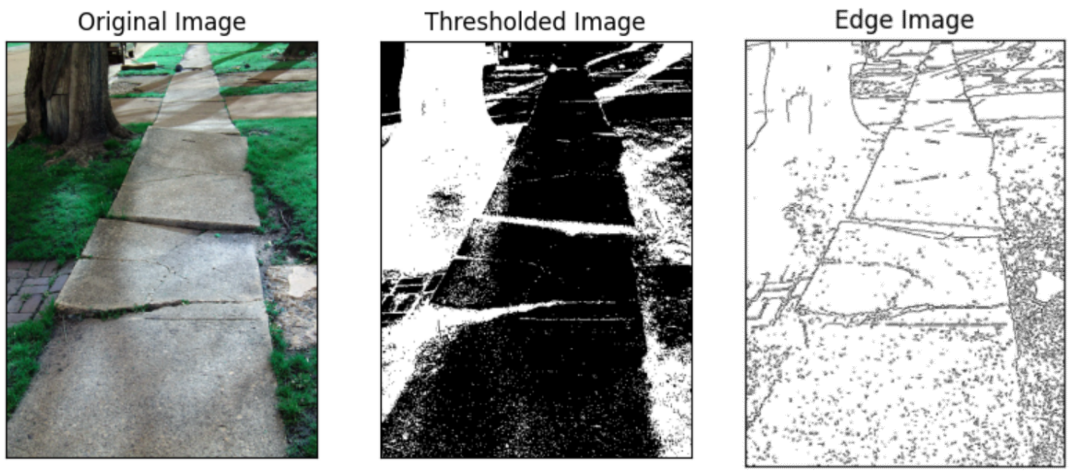
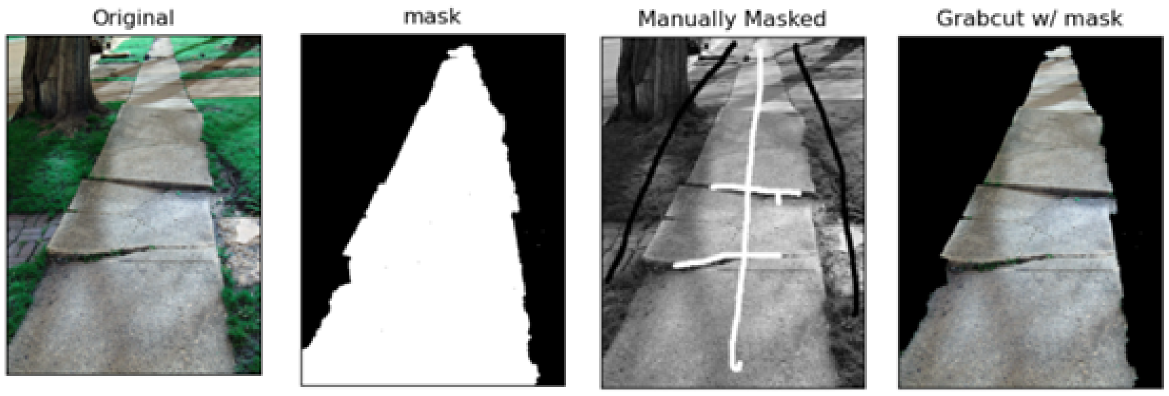
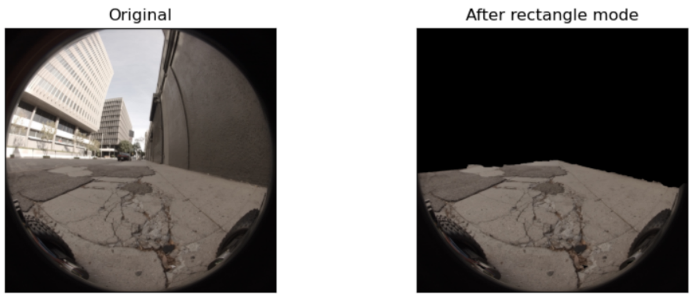


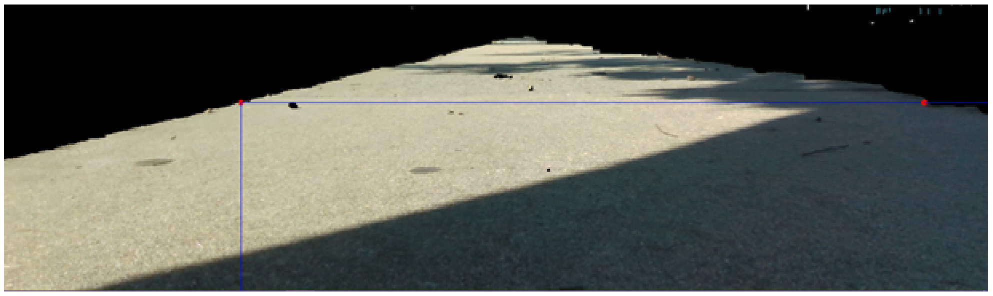
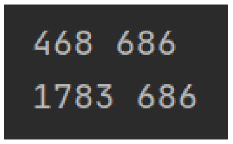
Image Segmentation and Grabcut Algorithm



Premasking and Grabcut Algorithm (Rectangle Mode) -Rover GoPro Camera



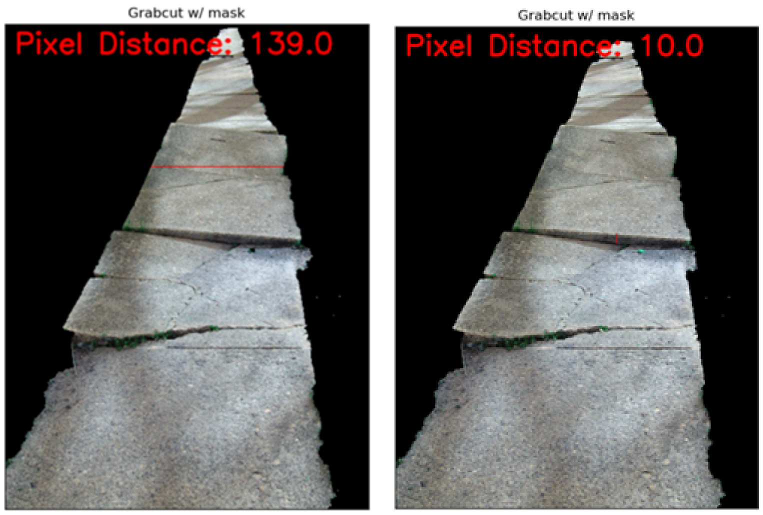
One goal of this task is to count the pixels so it can be scaled onto real unit measurement, specifically in centimeters. The above images can easily help find the edges of the sidewalk to precisely locate the starting and ending points for each of the x and y displacements. The following image shows how we extract the pixel coordinate of the image by clicking at one point on the image. Then, we can calculate the distance between those two points to find the pixel distance.

```
468 686
1783 686
```

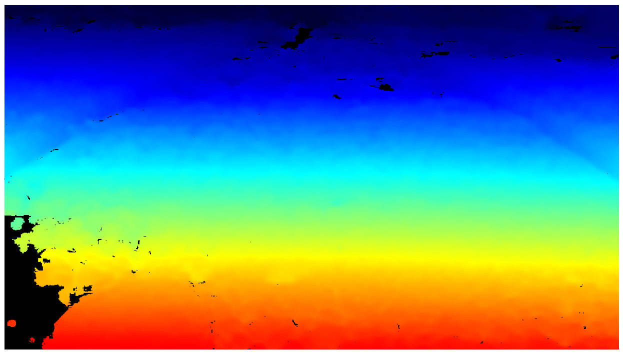
The ultimate goal is to get the measurements of the horizontal and vertical displacements on the sidewalk. Once we find the pixel distance for the x and y displacements, we convert it to centimeters. Those measurements will be used as part of the parameters in labeling the priority of the sidewalk images.

Horizontal Vertical



The Intel Realsense depth camera model d435 takes both colored and depth data. PNG files do not save depth data so saving as RAW is recommended. In this case, we saved some frames as a

BAG file to save the depth data. Next, using the Intel Realsense library, read the BAG file, take a frame, and then align both the colored and depth frames to map each pixel together. Once aligned, select two points and we can find its 3D coordinates from the depth data. Finally, perform Euclidean distance to find the displacement in meters and convert to centimeters. Displacement will be shown in the console.

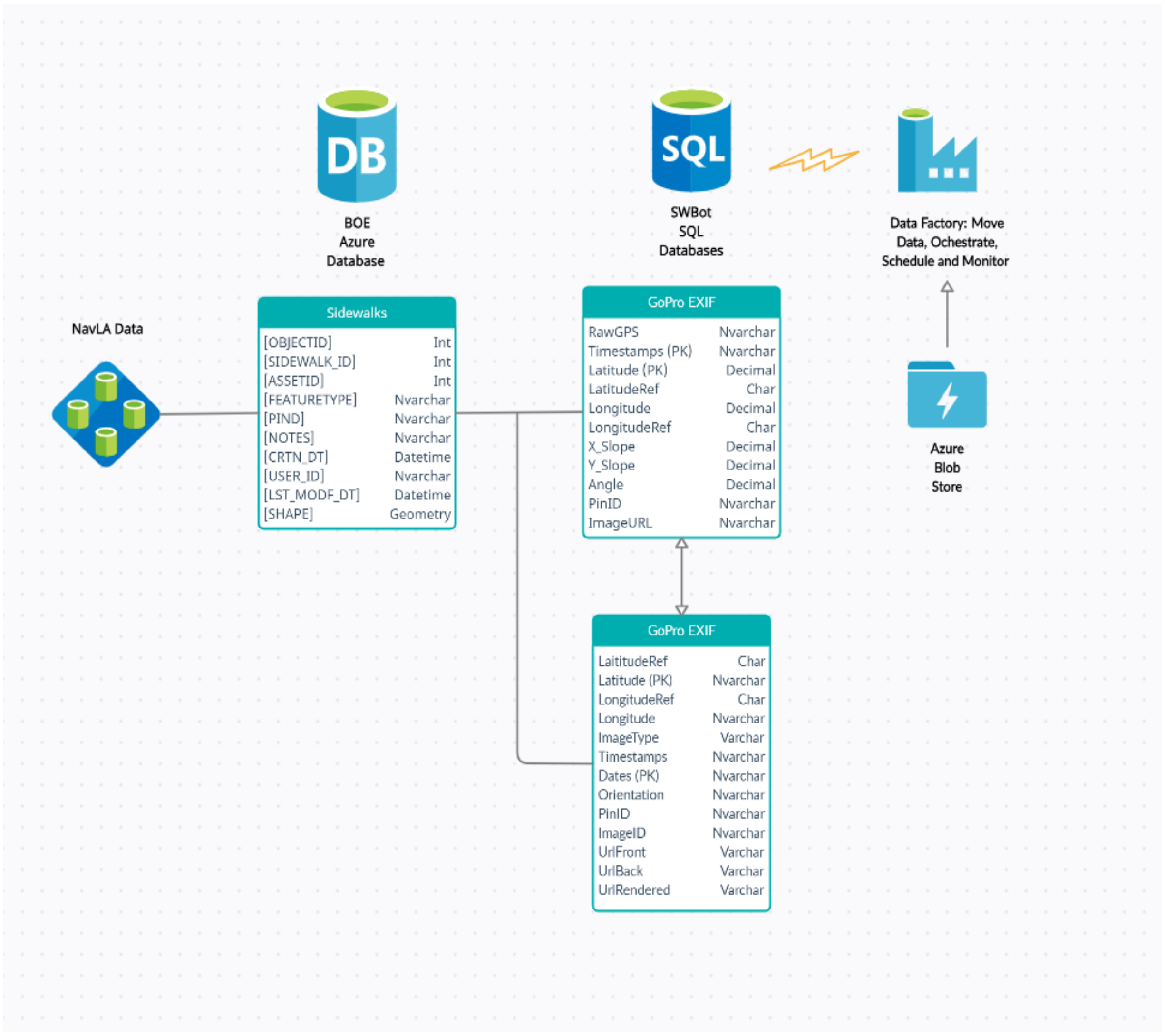


4.3 Task 3

- Rover Ui
    - The Rover UI will display all data that field workers have collected in one session
    - The data will be able to be saved in a csv file for field workers to analyze the data later on in the office
    - The Rover UI will control the movement and speed of the rover to collect sidewalk slope data.

4.4 Task 4

4.4.1 Database Data Diagram



Sidewalks

[OBJECTID]	Int
[SIDEWALK_ID]	Int
[ASSETID]	Int
[FEATURETYPE]	Nvarchar
[PIND]	Nvarchar
[NOTES]	Nvarchar
[CRTN_DT]	Datetime
[USER_ID]	Nvarchar
[LST_MODF_DT]	Datetime
[SHAPE]	Geometry

GoPro EXIF

RawGPS	Nvarchar
Timestamps (PK)	Nvarchar
Latitude (PK)	Decimal
LatitudeRef	Char
Longitude	Decimal
LongitudeRef	Char
X_Slope	Decimal
Y_Slope	Decimal
Angle	Decimal
PinID	Nvarchar
ImageURL	Nvarchar

GoPro EXIF

LaititudeRef	Char
Latitude (PK)	Nvarchar
LongitudeRef	Char
Longitude	Nvarchar
ImageType	Varchar
Timestamps	Nvarchar
Dates (PK)	Nvarchar
Orientation	Nvarchar
PinID	Nvarchar
ImageID	Nvarchar
UrlFront	Varchar
UrlBack	Varchar
UrlRendered	Varchar

5.0 Policies and Tactics

5.1 Choice of which specific products used

- Task 1
  - Web application
    - Django
      - Django uses Python for its backend. See Python under the mapping files bullet point to view why we select Python.
    - Google Maps API as a reference to iterate through images, similar to BOE.
  - Mapping files
    - AutoCAD
      - Command language available to automate mapping file creation. Forums and documentation thoroughly explains how command language may be used and provides sample use cases.
    - Python
      - Existing scripts from previous years efforts use this language. Python has an extensive library that can provide parsing and data processing abilities that is not readily available in other languages.
- Task 2
  - OpenCV Library
    - OpenCV library was chosen for image processing because it provides many functions that help with image segmentation.
  - Intel Realsense Library
    - Intel Realsense library was chosen for image processing because it provides many functions that help operate the Intel Realsense depth camera d435.
- Task 3
  - We choose to use Javascript and ROS due to the rover we acquired, Leo Rover, which uses a Raspberry Pi microprocessor. The database we choose is the Azure database based on our liaison, the city of los Angeles, using the same database. This is so we can transfer our data freely between the liaison and our project.
- Task 4
  - Rover
    - Leo Rover was chosen as a replacement for our previous rover "Robecca" because its more versatile, price friendly, and is a good platform a 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.

  - Database
    - Azure SQL database was chosen by our corresponding liaisons because it's what they 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.

- 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 pulling data from the Azure database and is not storing or using local data.
  - Mapping files
    - Automation scripts create DWG files that meet the design requirements. Design requirements include color schemes based on BOE's prioritization and scoring system, annotations for slope, etc.
- Task 2
  - Image Processing
    - Image Processing functionality is uploaded to one place when the task is done.
- Task 3
  - Web UI Application
    - The rover is actively saving data collected and is being stored in the local storage and Azure database
- Task 4
  - Backend Database
    - Database is constantly updated with unique data

5.3 Plans for testing the software

- Task 1
  - Web application
    - Users can view an image. The administrators can confirm that the image shown, and its pulled data are related.
    - Users can flip through the images and its related data is shown. See the first bullet point.
    - Users can auto flip through the images and all related data is shown. See the first bullet point.
  - Mapping files
    - When running the automation scripts, the automation scripts pull data from the Azure database and create DWG files.
    - The DWG files must display the shape of a sidewalk with annotations and any needed color schemes (applying color schemes based on BOE's prioritization and scoring system).
- Task 2

    - Image Processing
        - Constantly running various different images from the different cameras to test the program’s functionality.
- Task 3
    - Test the rover movement to simulate how field worker’s will collect data
- Task 4
    - Test data will be created and uploaded to the DB prior to going live.

- Quality Control

6.0 Detailed 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 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 MicroSd cards 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

6.2.6 Interface/Exports

The interface has been coded with python tkinter toolkit. The exports available will be a data CSV file.

6.3 Image Processing

6.3.1 Responsibilities

Image processing functionalities will be implemented for the images taken from the Lidar camera attached to the Rover. The functionalities include measuring the XY (vertical and horizontal) displacements on the sidewalk and eventually categorize the images according to their condition.

6.3.2 Constraints

If the quality of the depth frame is not good, such as it is grainy with black spots, then the depth cannot be found in that specific area.

6.3.3 Composition

We use OpenCV or Open-Source Computer Vision library that is compatible with Python for image processing functionality. We currently use Pycharm IDE to utilize those components.

6.3.4 Uses/Interactions

If the strokes for Grabcut technique mentioned earlier cannot be implemented automatically, user strokes are a requirement to isolate the sidewalk, when measurements are made.

6.3.5 Resources

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

6.3.6 Interface/Exports

OpenCV library built in GUI with input and processed image.

6.4 Web application and mapping files

6.4.1 Responsibilities

The web application is a console for users to view a single instance of a GoPro image and its related data. The mapping files script will create DWG files automatically that can either be imported or hosted on NavigateLA. The DWG files will contain data from the GoPro image, too.

6.4.2 Constraints

Web application 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 create DWG files.

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, image processing scripts

6.4.6 Interface/Exports

Web application is the web console and the mapping files scripts create DWG files.

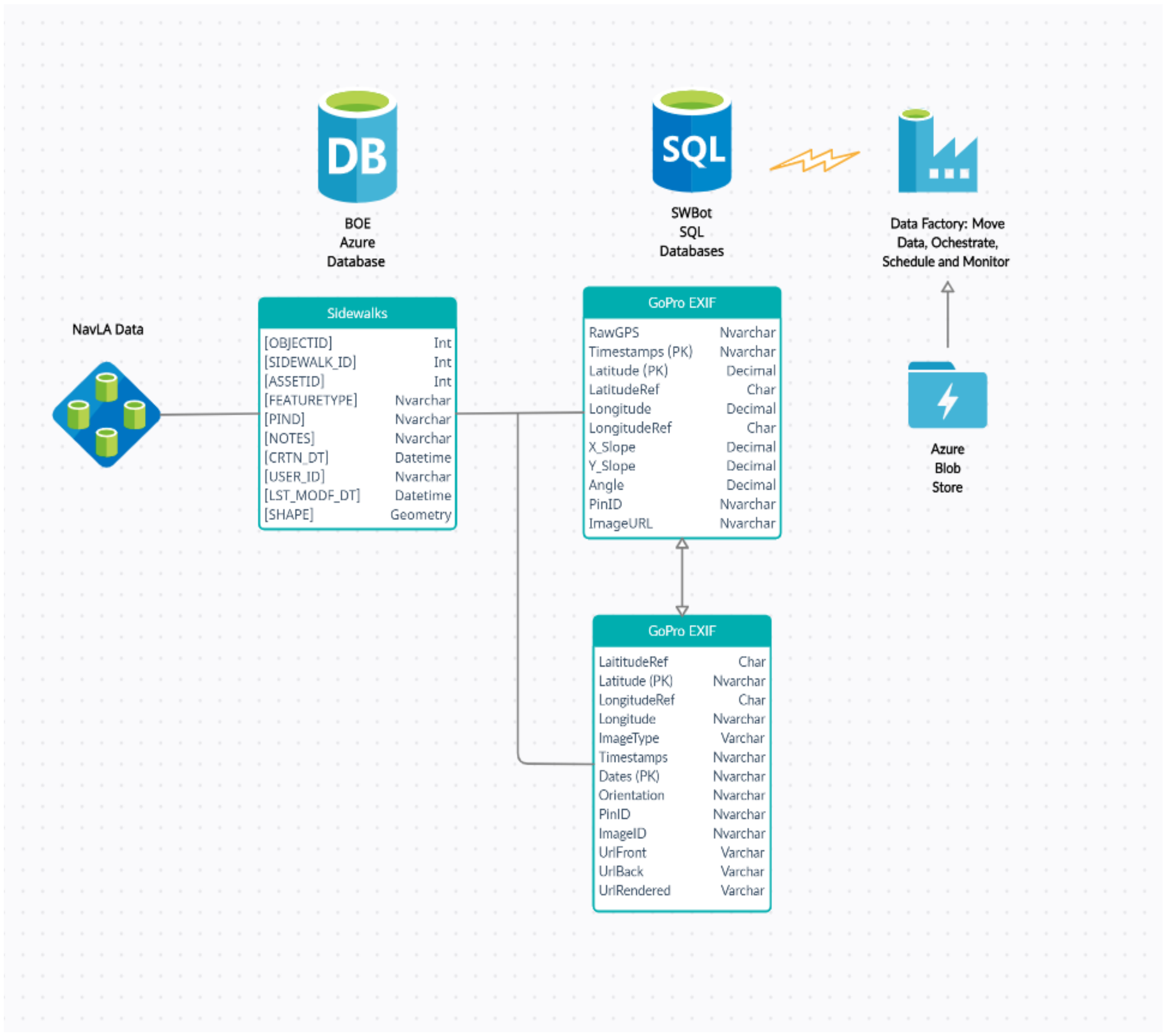
7.0 Detailed Lower level Component Design

To be determined in Spring

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



Sidewalks	
[OBJECTID]	Int
[SIDEWALK_ID]	Int
[ASSETID]	Int
[FEATURETYPE]	Nvarchar
[PIND]	Nvarchar
[NOTES]	Nvarchar
[CRTN_DT]	Datetime
[USER_ID]	Nvarchar
[LST_MODF_DT]	Datetime
[SHAPE]	Geometry

GoPro EXIF	
RawGPS	Nvarchar
Timestamps (PK)	Nvarchar
Latitude (PK)	Decimal
LatitudeRef	Char
Longitude	Decimal
LongitudeRef	Char
X_Slope	Decimal
Y_Slope	Decimal
Angle	Decimal
PinID	Nvarchar
ImageURL	Nvarchar

GoPro EXIF	
LaititudeRef	Char
Latitude (PK)	Nvarchar
LongitudeRef	Char
Longitude	Nvarchar
ImageType	Varchar
Timestamps	Nvarchar
Dates (PK)	Nvarchar
Orientation	Nvarchar
PinID	Nvarchar
ImageID	Nvarchar
UrlFront	Varchar
UrlBack	Varchar
UrlRendered	Varchar

8.2 Diagram Description

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

The left most part, being the existing data provided by BOE: sidewalk. It includes PIN ID's and asset ID's which are essential in pinpointing specific properties and sidewalk locations. Navigate LA data will serve as geographical reference points for the Image data and the rover data listed.

The right data tables, the Rover Data and GoPro EXIF tables would be extracted from the rover itself--using the timestamps and PIN ID columns to link it to the Image data Table.

The correlation between the location (longitude and latitude) points provided by both the rover/GoPro and the NavLA data can be accomplished through Azure.

8.3 Azure blob Storage implementation

Azure Blob Storage is required to store a large amount of JPG's--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. Blog storage will also be used to store CSV files of our data so we can automatically upload our data once collected from the memory cards.

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.
4. Azure tools and other NavLA tables will be implemented. As well as automation at every step of uploading data.

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.

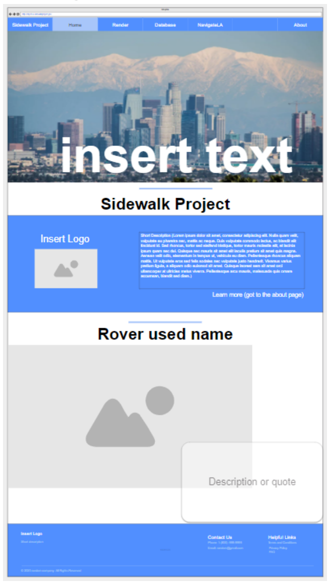
Our web application will include a page to display our data. The data to be displayed includes images received from the rover, longitude and longitude slope data, Global Positioning System (GPS) data as well as the image name and date when what image was taken. The web application will have the ability to iterate through the image with a previous button, next button, and an auto button. The auto button will literate through the images automatically, with each image showing for 2 seconds. The user will be able to pan the image and zoom in and out of the image. The web application will have a function to search through the database of pictures by the images name or coordinate which the image was taken from.

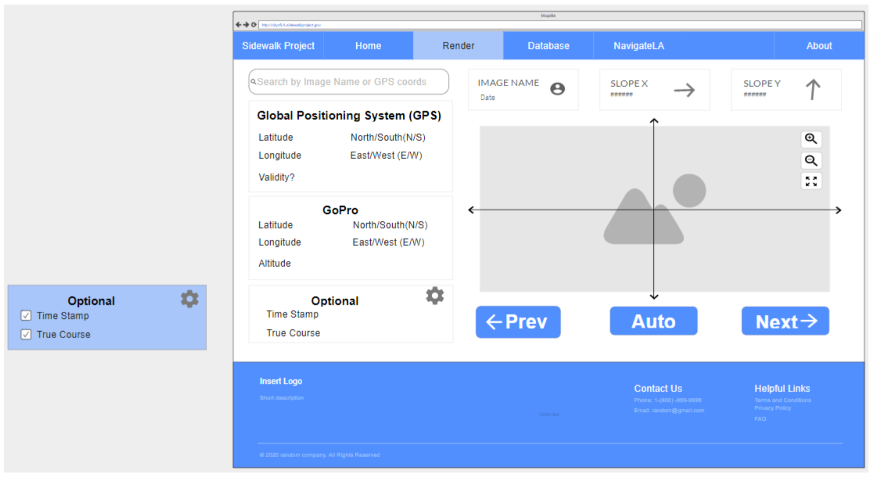
The web application will have additional pages which will describe the purpose of our project, the overall description, the environment where our project was worked on, and the algorithms used in this project.

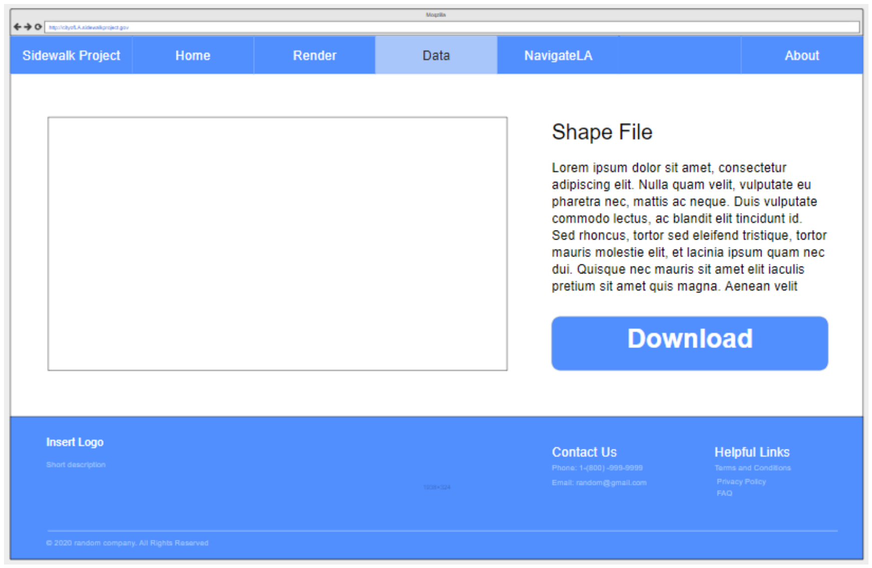
9.2 Screen Frameworks or Images

9.2.1 Task 1, Web Application

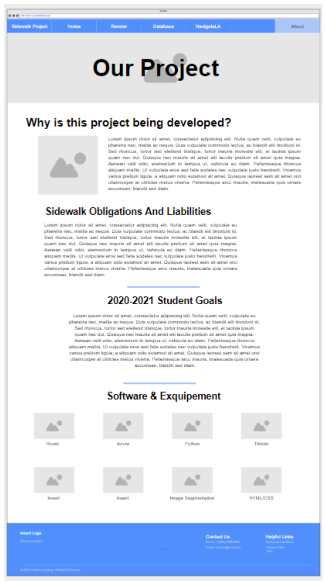
The following are wireframes and actual frontend images:

- 

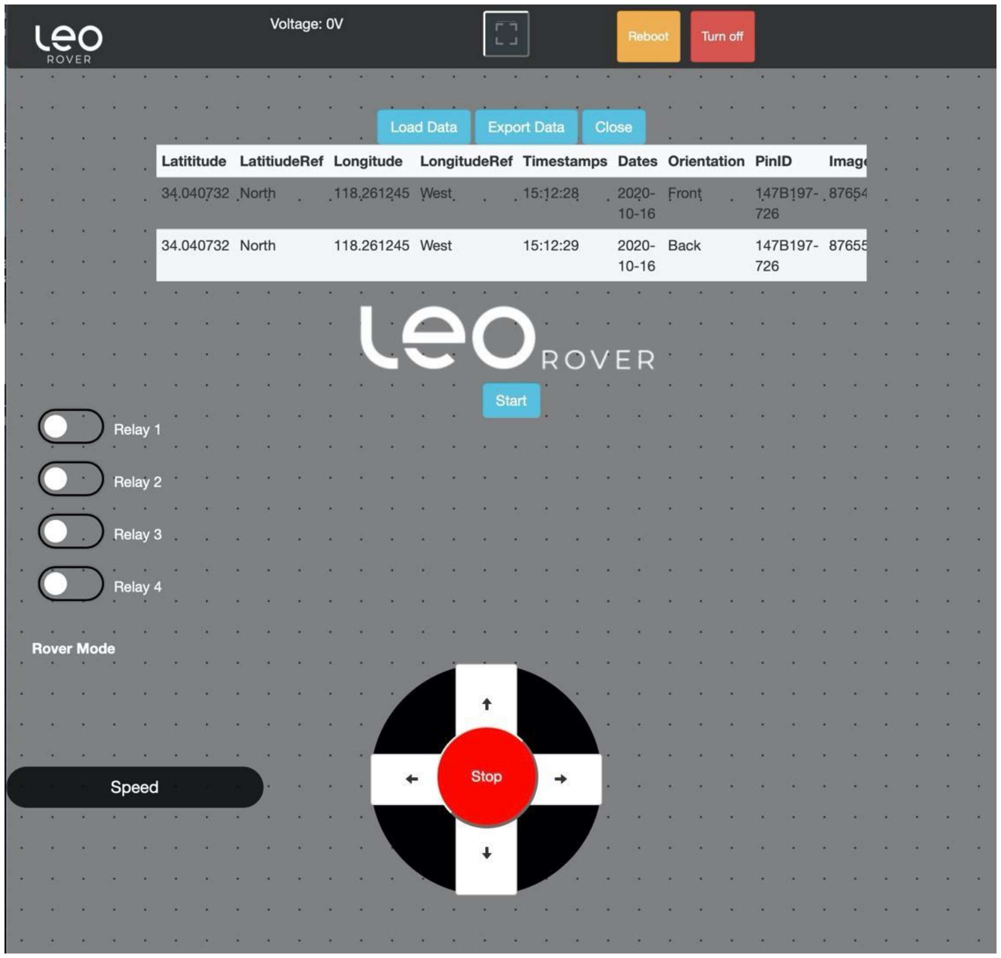
- 

- 

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui. Quisque nec mauris sit amet elit iaculis pretium sit amet quis magna. Aenean velit

- 

9.2.2 Task 3, Rover UI

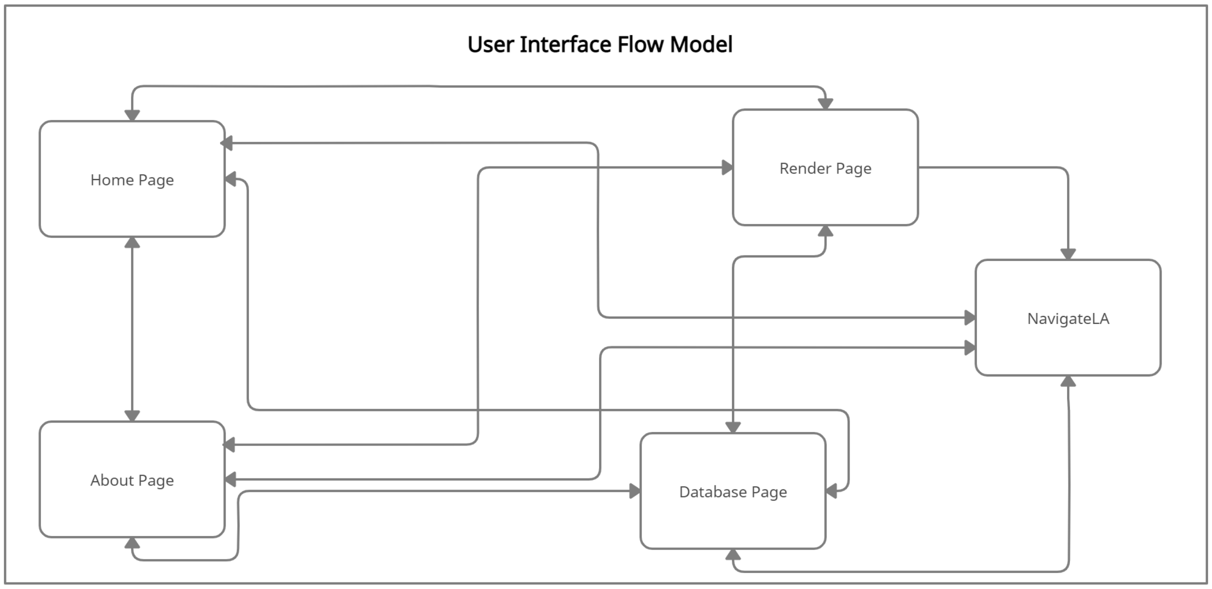


Latititude	LatitiudeRef	Longitude	LongitudeRef	Timestamps	Dates	Orientation	PinID	Imag
34.040732	North	118.261245	West	15:12:28	2020-10-16	Front	147B197-726	87654
34.040732	North	118.261245	West	15:12:29	2020-10-16	Back	147B197-726	87655

9.3 User Interface Flow Model

9.3.1 Task 1, Web Application

The Home, Render, Database and About page can freely navigate between each other. However, the NavigateLA tab is different. It takes the user to a different webpage, https://navigatela.lacity.org/. Once at NavigateLA, the user cannot navigate back to the web application, unless the user presses the back arrow on the user's web-browser.


User Interface Flow Model
Home Page
Render Page
NavigateLA
About Page
Database Page

10.0Requirements 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

11.0 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	Image processing
Task 3	Rover UI
Task 4	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