Real Time Route Optimization

DESIGN DOCUMENT

Team Number: 18
Client: Adam Ryan
Adviser: Goce Trajcevski
Team Members/Roles:

Junjie Wen - Backend Developer; Data Analyst
Zhanghao Wen - Product Manager; Scribe
Yuhang Xie - Web-App Developer; UI Leader
Xinhe Yang - Web-App Developer; UI Designer
Tianhao Zhao - Communication Leader

Team Email: sdmay19-18@iastate.edu

Team Website: http://sdmay19-18.sd.ece.iastate.edu/

Revised: 11/30/2018 Version 2

Table of Contents

List of figures/tables/symbols/definitions

1. Introduction

The purpose of this project is to develop a system for effective management of a fleet of snow-plowing trucks.

1.1. Acknowledgement

Our client, Mr. Adam Ryan from Henderson Products Inc., provides us with the overall guidance about the different facets of the problem. In addition, he will provide us with the AWS and database runs from the different sensing devices on the trucks for the purpose of enabling a testing of our solutions. Besides, he guides us through the process of representing (i.e., pre-treating) the data on the map with trucks in snow operations, to help route arrangement of snow plows. For all of this, we are very grateful.

We thank to our adviser, Professor Goce Trajcevski, who provides us with valuable advices on our decision and planning throughout the semester. He helps us making decision on how we can approach the tasks in multiple aspects, how we can go through our thinking process of finding main cores of objects in designing, and how we could overcome current challenges with overall thinking of the project.

1.2. Problem Statement

After snowfall, it is important to clear both road segments as well as driveways, so that: (a) travelers safety is improved, and (b) safe access is enabled for employees, customers and suppliers. Car-slip accidents are quite common during winter seasons, especially in North America. Thus, an effective management of snow cleaning/removal is of the utmost societal importance – however, snow cleaning operations could incur high cost, both in terms of human labor and financial resources. Our client, Henderson Products Inc., is a custom manufacturer who ensures that their customers (including state departments of transportation, cities, counties, and other governmental agencies) have the most complete snow and ice fighting solutions and service available ("HENDERSON PRODUCTS - ABOUT US"). Presently, Henderson Products Inc. is seeking a solution which takes advantage of modern Internet of Things and Big Data paradigms and technologies to improve the operational performance of snow-plowing fleets of trucks in and reduce their service and re-loading time as well as financial costs. At the current state of the affairs, the snow operation dispatchers or the target users of our project, may use some specific ways to collect data in order to get the status of the trucks location, dispenser info, and road conditions during snow operations - however, the data collection, analysis and decision-making are done in a rather ad-hoc manner. Hence, our team is aiming at providing a systematic approach and generate a solution that will allow them to have a better experience of gathering real-time information of trucks and controlling the real-time situation of arrangement of fleets during snow operations. Our solution is intended to enable dispatchers to perform a more effective decision-making in terms of tasks re-assignments and trucks re-routing (both in terms of road segments to be cleaned, as well as reloading with supplies and servicing).

1.3. Operating Environment

Our final product is a web application and it will be used by the dispatchers as well as managers in certain branches/office of Department of Transportation (DoT). Since it is a web-based design, the hardware such as equipments of sensors are not connected to our design, the actual physical environment will not be directly relevant to our envisioned product from development perspective, except a snow weather condition for utilizing our product. It will require the use of specific local and online system with accessible data that fits and operates the software solution from our design.

1.4. Intended Users and Intended Uses

Our project Real-Time Route Optimization aims at optimizing the routing of the snow plow trucks and reduce the resources wasted during operations. There are three types of users, each with different access interface: (1) general public; (2) dispatchers for arranging trucks; and (3) operator/manager/admin at DOT office, who will have a full access to different information, along with the privilege to come up executive re-routing schemes.

For the public view, citizen can access our website to find out the nearby snow plowing trucks which displayed on the map, as well as the status of which road segments have been cleaned.

The dispatcher's task is to assign the route for directions of each snowplow, and maybe able to show status of truck supplies (e.g. melting salt, dissolved liquid, gasoline). In this process, the snowplow may need to pass the same path and redo snow operations due to bad road conditions. Our project is to help dispatchers properly distribute the tasks of snowplow and reduce waste of resources.

Admin level users, such as office manager, can see the situations during re-routing process, monitor the tasks completion by dispatchers, or help solve unexpected conditions. They may get records of data, do some high level analysis on specific areas of road for future preparation.

1.5. Assumptions and Limitations

Assumption :

- Users will have internet access
- Users will have a computer system that deal with large amount of data, and be able to hold a large number of caches for storing trucks info on the page during a certain time period with real time updating of info.
- The maximum number of simultaneous users will not be limited.
- The users will be able to read instructions on our design of representing the map with trucks and road information.

Limitations:

- Because we are not able to use real time data collected on sensors for developing our design, we need to build up a simulator that conclude all data types of requirements, for future connection with real data purpose.
- Some features of our design has been limited, such that we are not confirmed that we are able to access camera info of the street, or some administration permit from the governor.
- The amount of running cache depends on users' computer system info. Better computer system will have better experience of surfing our map page.

1.6. Expected End Product and Other Deliverables

A web application map with single or multiple interfaces mainly designed for three types of group. The first group is for public. Citizens can access our website to find out the nearby snow plow which displays on map as well as which road has been cleaned. The second group is dispatcher. They can know real time road conditions and do re-routing arrangement to optimize snow operations. The third group as admin managers, have full access to control and monitor fleets during snow operation.

A MySQL database which stores all types of data about truck information and users' information. Data could be queried by AWS to monitor and analyze performance of fleets.

The web application could get data from the server with same data types in backend simulator, and provide detailed truck and road information.

2. Specifications and Analysis

We now discuss in details the requirements specifications and design analysis.

2.1 Functional Requirements

- A web application, which contains:
    - Trucks' information along with movement
    - Road conditions
        - color specified conditions determine whether cleaning is needed.
    - Temporary data of each truck along the path
        - it depends on web caches capacity
    - Home page
    - Login page with multiple layers of users
    - History page to store past data
- A mobile application, which contains:
    - Trucks' GPS location on map
    - Individual truck's supply condition
    - Company's name
    - Communicate with operation center
- AWS server implemented by Nodejs, which contains:
    - Simulator to generate truck data for display, since we cannot get real data
    - Authentication for driver/company to login
- MySQL database, which contains:
    - Table for truck gps and support condition data
    - Table for user information

2.2 Non-Functional requirements

- Web application --- physical data requirements:
    - Trucks' information
        - GPS locations
        - Rate of water/salt dispenser
        - supply condition (if applicable)
    - Road conditions
        - traffics with traffic speed
        - road temperatures
    - Weather conditions
    - Camera access on the road (if applicable)

- Contents in App and Web interface design should be easy to read and understand

2.3 Standards

The team will use the following standards during development of the project:

- Version Control System
    - Git will be used as the primary means of version control for all project code.
    - The Google software suite (Docs, Sheets, etc.) will be used for all formatted documentation, such as planning and design documents. It has version control features that can accessed from the menu (File → Version history).
- Code Review
    - Development will be done in feature branches.
    - When a feature branch is ready to be merged to the master branch, the author will assign one or more of the other project members as a reviewer.
    - Reviewers must check to ensure the code:
        - correctly implements the desired functionality
        - contains enough tests to ensure correctness
        - passes tests
        - is free of errors
        - integrates successfully with the existing software
    - Reviewers will communicate code issues to the author, who is then responsible for addressing all issues.
    - Once the code has passed inspection, it can be merged to master.

2.4 Proposed Design

For implement the algorithm of re-routing, the following information should be gathered and considered: traffic condition, weather condition, rate of water/salt dispenser, and GPS information of individual truck during snow operations. Therefore, the following tools, API, and framework are used for the purpose of front-end interfaces, backend database and server.

2.4.1 Web application

We choose to use AngularJS as our web design interface framework. It has a good html block handling. AngularJS is a Google product, so it has good compatibility with Google map api. In comparison to ReactJS which lacks official documentation, it has a lot of information to help beginner to learn and easy to use with simple html knowledge. It accepts other frameworks with similarity in architecture design. It also has tiny size which allows faster and better performance comparing to ReactJS. It also has a great integration with small interactive parts. This is a lot better than ReactJS which requires deep knowledge of integration. We can simply apply the single application to complicate web interface applications by using AngularJS .

2.4.2 Database

Development: MongoDB is friendly to developing engineer, because of its supply of JSON format data. With the Mongodb, developing speed can get improved. For MySQL, it is a more mature solution. There are lots of documents about MySQL. Besides MySQL is also extendable for different data type in the future. So, for development, MySQL and MongoDB does have significant advantages than the other one.

Maintenance: relational database is very good for maintain because the rules in operation on table. On the other hand, no-relational database is not easy to maintain, and more risk in illegal operation on database. So, for maintain, MySQL is also friendlier to engineer.

Load: Compared with MySQL, MongoDB is better on loading.

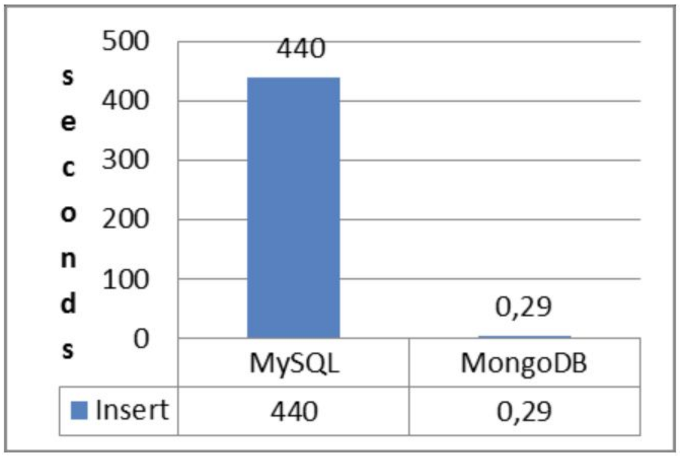


Figure 1: MySQL vs MongoDB

2.4.3 MAP API

We are considering several map APIs including Leaflet, Bing Map, and Google Maps.

Leaflet is a widely used open source JavaScript library used to build web mapping applications.

Bing Maps is a web mapping service provided as a part of Microsoft's Bing suite of search engines and powered by the Bing Maps for Enterprise framework.

Google Maps is a web mapping service developed by Google. It offers satellite imagery, street maps, 360° panoramic views of streets, real-time traffic conditions, and route planning for traveling by foot, car, bicycle, or public transportation.

2.4.4 Block Diagram

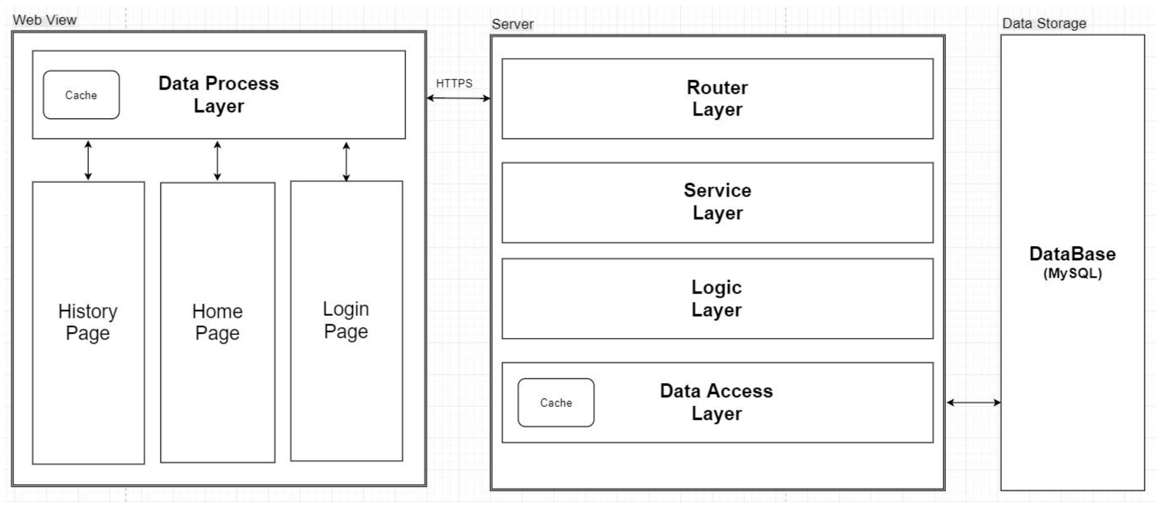


Figure 2: High Level Architecture Diagram

2.4.4.1 Web View

Data processing layer takes the simulated data (such as location or speed of truck) from the server temporarily store the data in cache and then logically transfers the data to display page. In our case, the truck's simulated real-time data is stored in the cache as an array, and the data processing layer displays the data on Google Maps. The login page requires user to enter username and password to login. The data is compared with the database data through the data processing layer to determine whether the data is the same. The home page mainly displays information on the google map, and the icon shows the specific location and information of the truck. The history page requires the user to enter the time range. The data processing layer will get data of database, display truck's driving route and information on Google map, and display the data analysis of truck during this time, such as average speed and total mileage.

2.4.4.2 Server

1. Router layer used to assign different pages and display different content data. In our case, we can access login page, home page and history page through the router layer.

2. Service layer used to simulate truck data such as GPS position, speed, bearing and spinner speed.

3. Logic layer will get data for logical processing, it stores the truck's simulation data in the database or calls the data needed by front end from the database. It will determine

if the password entered by the user is consistent with the database. If it is inconsistent, it will return error message to the front end. It can analyze the data obtained by the database and calculate the average speed or total mileage of the truck.

4. Data access layer temporarily store data in cache, waiting for other components to call the data

2.4.4.3 Data Storage

Classify data and store it in the database or take data from database. The user's username and password are stored in the UserTable, and the user's level is divided into normal users and administrators. The truck information is stored in the TruckTable and contains fields such as Altitude, GPS_head, Acc_mag, Conveyor, Spinner, Prewet and Time.

2.5 Design Analysis

2.5.1 Software Tools Specifications

The web application is going to be developed using JavaScript, with the Vue.js framework used on the client side and Node.JS used on the server side.

The mobile application is going to be developed with the Ionic framework used on the client side and Node.JS used on the server side, our database is s MySQL database.

Although native apps can take advantage of OS features and other software tools that are installed on that platform to maximize its performance, we choose cross-platform mobile framework in our app design.

The Ionic framework uses web technologies like HTML5, CSS, and JavaScript to write and run applications, and requires Cordova wrapper to access native platform controllers. The Ionic core is written with Sass and was originally based on a popular JavaScript framework – AngularJS.

The Ionic main programming language is TypeScript, which is generally a superset of JavaScript that compiles to plain JavaScript. TypeScript increases the quality of the code because it helps to spot and eliminate mistakes during code typing. Using TypeScript is optional, and the application can be written in simple JavaScript.

The core reason using Ionic is that it provides us high efficiency of coding since the percentage of code reuse among all other cross-platform mobile frameworks is the highest. Although performance statistics is relatively low compared to others, it is acceptable in our project.

The reason we choose MySQL is in our project, maintaining is more important than load. Because in this project, the users will only be the snowplow driver and the dispatch center, so there will not huge load. And since most engineer will leave the project after next May. Whether it is maintainable is very important.

We chose Node Js instead of Spring framework for the following reasons:

- NodeJs is a lightweight background framework with good scalability, ideal for real-time data interaction applications.
- NodeJs server configuration is easier and faster than spring.
- NodeJs is developed based on the javascript language, it is easier to get started, more suitable for web development.
- Significantly reduce the amount of code developed so improve development efficiency.

Spring is better than Node JS in overall performance. However, we need more time, funds and people to develop a sophisticated project by using Spring. Under these premises, if we use spring as the back-end development framework, it will extend the time of our project delivery.

Google Map Api is used because:

- Google map API can handle a huge number of markers (10000+) at same time and it is more efficient than other map APIs. Our product needs to handle many markers at same time and each of them represents the position of one truck. Many markers mean that when the map is zoomed out, the user cannot clearly see how many trucks are in a certain area. But the google map API has a Marker Clustering feature. It can show the number of markers in a certain area when map is zooming out.
- Google map is friendlier to new developers. The team members in our group did not have the experience of embedding maps in web pages and mobile software, so we need a map API which is powerful and easy to use. And google map has all the features we need for our project, such as geolocation, autocomplete boxes, traffic display, position markers and so on. And the most important thing is that these features don't depend on any third-party plug-ins, which means we don't need to spend more time learning new software and evaluating the quality and performance of them.

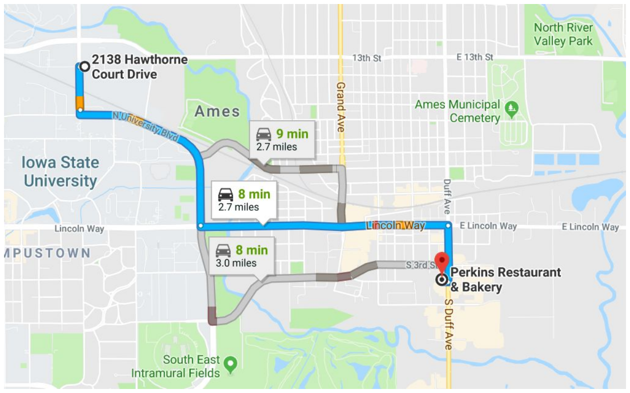
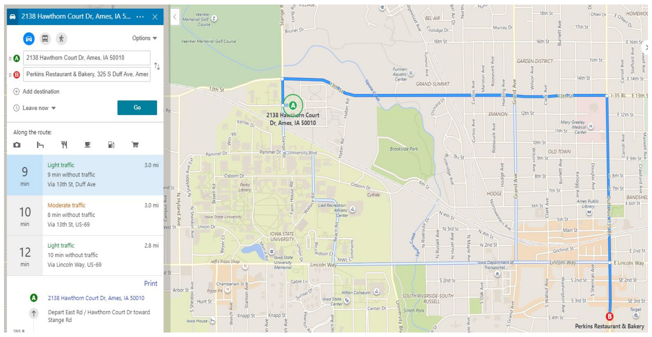


Figure 3: Google API (1)

- The google map API has a custom layer feature. It can be customized to include adding and removing place names, changing the color of various features such as display red or orange line depending on how crowded the road is. Our product can use this feature to distinguish between roads that have been cleaned and roads that have not been cleaned.
- It is easy to import data into Maps. For our project, we need to get the location of each truck from AWS and update it on the map in real time, so this feature is very useful for us.


2138 Hawthorn Court Dr, Ames, IA 5...

Options

A 2138 Hawthorn Court Dr, Ames, IA 50010

B Perkins Restaurant & Bakery, 325 S Duff Ave, Ames

Add destination

Leave now

Go

Along the route:

9 min — Light traffic — 3.0 mi — 9 min without traffic — Via 13th St, Duff Ave

10 min — Moderate traffic — 3.0 mi — 8 min without traffic — Via 13th St, US-69

12 min — Light traffic — 2.8 mi — 10 min without traffic — Via Lincoln Way, US-69

Print

A 2138 Hawthorn Court Dr, Ames, IA 50010

Depart East Rd / Hawthorn Court Dr toward Stange Rd

285 ft

2138 Hawthorn Court Dr, Ames, IA 50010
Perkins Restaurant & Bakery

Figure 3: Google API (2)

- Leaflet: Adding a map to a simple website is a lot trickier for the average person using Leaflet. Many features will prove too much effort for those looking for a plug-and-play solution.
- Bing map: From two figures we can clearly see the difference between the two map APIs. First, the recommended route for the Bing map is not the most time-consuming. Secondly, it does not show the congestion of the route, which makes the user unable to select the appropriate route. The most important thing is that our final product also

includes a mobile app, and the Bing map hardly supports mobile whether it is IOS or Android.

As we move forward, we will continue to build out the services and functionalities for both our app and web. We will continue to build out the AWS server and database functionality through a more extensive needs from front end website and mobile application.

2.5.2 Design Specifications and Analysis

*Parts of design analysis are also the implementation issues as our design revisions.

1. Thinking Process
    a. Originally, we thought we are going to implement a software solution to arrange snowplows in real time, which may require complicated algorithm upon the database from server.
    b. After talking to client, we had a big decision change made on our progress. We move our emphasis to frontend as adjective or auxiliary system of displays, to display the info needed for dispatchers from snow-operation office.
    c. Then we need a simulator which can be used to generator data to support frontend and be connected to real data with similar forms of data type.

2. List of main design objectives
    a. Frontend --- Login Page
    b. Frontend --- Main Page
    c. Backend --- Simulation

3. Frontend --- Login Page
    a. Thoughts
        i. We need to have a login button.
        ii. We also need place to put username and password
    b. Problems
        i. The model of router cannot be splitted into multiple pages.
            1. Because there is only one model of router.
            2. We add multiple layers of model in order to make it possible to display multiple pages of corresponding contents.
4. Frontend --- Main Page
    a. Thoughts
        i. Initially, we want to display a truck or car on the map.
            1. Import Google API
            2. Find a marker in Google API.
        ii. In order to make it move
            1. Import data for the truck
                a. Create a backend server
            2. Update data frequently and move trucks on the map

        3. Add multiple trucks.
    iii. We want the truck to move smoothly
        1. We divide the single movement distance into several small segments and display it.
        2. It will let users have better experience of using our product.
    iv. We want to show the trace of the truck
        1. We use polyline to display the segment of the movement
    v. We want to temporarily store info of the truck for further implementation use
        1. We add slots on the path the truck passes to store infos.
        2. The slots are added and disappeared in certain time sequences.

b. Problems

(As 1.Analysis/Thoughts
2.Solution)

    i. The truck jumps to the other coordinate for each movement
        1. We want it to move smoothly, but it jumps to the other coordinate.
        2. We limited the range of data generator to let trucks move in small range. We divide the single movement into several small segments on its distance, in order to make it seems to be real for simulation.
    ii. The info window of the truck shows only during the time before its update. After each update (few seconds), it disappears.
        1. We want to click once and shows it for a long period of time.
        2. We add a variable to the initial frame of updation and keep its property of last time of whether the info window is open or not.
    iii. The movement of truck shows on the page sometimes get stuck due to it produces a lot of virtual memory.
        1. We want the program to run for a long period without stuck.
        2. Use First In First Out algorithm to keep the group of data in the same size. This method can ensure that the memory will not overflow and cause slow operation.

5. Backend --- simulation
    a. Thoughts
        i. Originally, we thought we need a backend server in order to make connections among database, frontend, and backend algorithms.
        ii. For backend algorithm, we need a simulator to generate the info corresponding to all real data types, in order to motivate frontend page.
        iii. For changing some of frontend features, we need to change a little bit of backend algorithms.
    b. Problems
        i. Overlay occured when we extend our display of one truck to multiple trucks

            1. Multiple trucks shows at one location instead of multiple locations.
            2. We redesign the data generator in the simulator.
    ii. NGINX --- using reverse proxy

c. Technical Challenge
    i. Learn JavaScript in order to build Node.JS based backend server.

d. Our decision
    i. We determine to build the data simulator as a microserver, to improve the extensibility of overview architecture.

3 Testing and Implementation

3.1 Interface Specifications

The final product can be used on two platforms.

Mobile app: Our "LetItGo" app includes information for driver such as company name, dispatched road for driver (also displayed in map), supplies of truck. App is designed and implemented by Ionic framework.

Web: It will be implemented by the vue.js framework. The interface mainly includes a map and a navigation bar. The map will dynamically display the route of all trucks within a certain area. Users can access the truck information interface through the navigation bar which includes the truck speed and supply information. We will test the web interfaces with real-time data.

3.2 Hardware and software

3.2.1 Hardware testing

The sensor data generator provides us with simulated truck data. These simulations can simulate data changes in the truck's driving process, such as GPS location, weather, and more.

3.2.2 Software/framework testing

We considered using spring or Node.JS to build our backend development framework. Finally we chose nodejs Here is an example of "Hello World" program by using Node.JS and Spring:

Obviously, Node.JS can reduce tons of code compared to Spring.

On the software side, we will use vue.js to develop the web to provide a line system for trucks. The simulated sensor data obtained is analyzed and algorithm calculated. Display the real-time location of the truck and the data in the car on our web.

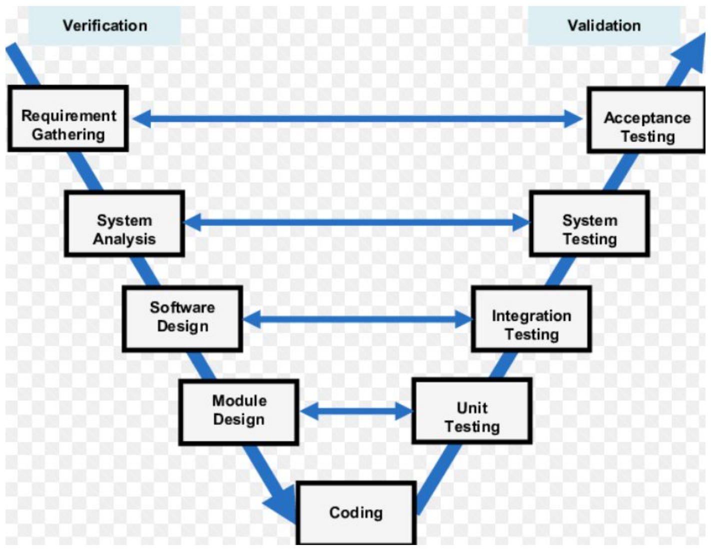


Figure 5: Layers of Testing ("Filling the Gap between Conception and Unit Test.")

3.3 Functional Testing

We will do functional testing based on the functional requirement, by using the simulating data generator.

Tested requirement:

Display trucks' GPS location on map as well as traffic condition

Expected result:

simulating data generator will send GPS location of trucks, and user should be able to see how truck move on the web.

Tested requirement:

Check out individual truck's supply condition (rate of water/salt dispenser)

Expected result:

simulating data generator will send truck's supply conditions of trucks, and user should be able to check all relative information on the web by clicking the truck.

Tested requirement:

Allow to open weather condition on certain snow operation area.

Expected result:

User will click the weather button, and see the snow on certain area.

3.4 Non-Functional Testing

For non-functional test, we test our project based on stress testing and performance test.

Stress testing:

We send tons of request to our server by JMeter, and test the limitation of backend server to see if it can handle enough load.

Performance test:

We repeatedly use JQuery to get the details information of how the front end page be load. And analyze if the front end page have good performance. And for backend server, we also using JMeter to send multiple type of request, monitoring the performance of server.

3.5 Process

Our planning of tasks shows as Figure 5 below.

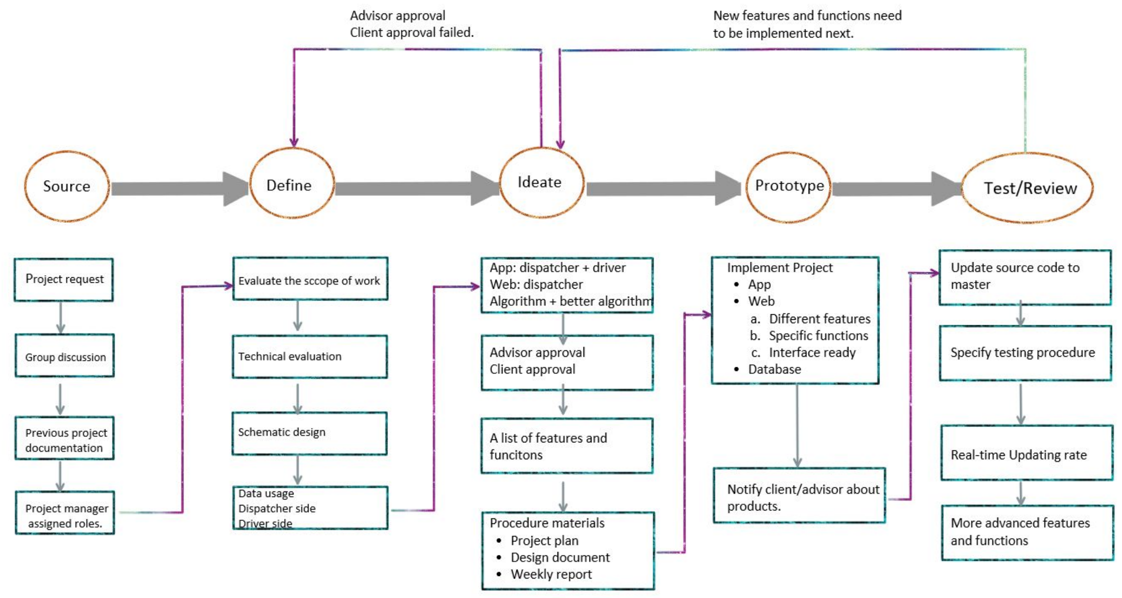


Figure 6: Design Thinking Process

3.6 Implementation Issues and Challenges

This sections is for issues encountered by designing implementation, since we haven't done any of test yet. This is also a part of analysis of our design thinking process.

3.6.1 Frontend Implementation Issues

- Login page

  The issue is that the model we used at first can only handle a single page. However, client proposes that the program needs to be able to provide different functions based on users' privilege which means we need a login page.
  The solution is to replace the module we originally used with the router module. Router module can place each page in different layers, so that the program can contain multiple pages with corresponding contents.

- Main page
  1. Our goal is to make the truck move along the route smoothly. However, because of the problem of page refreshment, trucks jumps to the other coordinate for each movement. In order to solve this problem, we limited the range of data generator to let trucks move in small range. We divide the single movement into several small segments on its distance, in order to make it seems to be real for simulation.
  2. After completing the function of displaying vehicle information on the map, we encountered a new problem - The info window of the truck shows only during the time before its update. After each update (few seconds), it disappears. However, our expect result is to click the truck once and shows it for a long period of time. Therefore, we add a variable to the initial frame of updation and keep its property of last time of whether the info window is open or not.
  3. Our expect result is that program can run for a long period without stuck. However, the movement of trucks shows on the page sometimes get stuck due to it produces a lot of virtual memory.To solving this problem, we had tried a lot of algorithms and and compared their effectiveness. Finally we chose First In First Out algorithm to keep the group of data in the same size. This method can ensure that the memory will not overflow and cause slow operation.

3.6.2 Backend Implementation Issues

When we tried to display a truck to show multiple trucks, we encountered a overlapping problem. Multiple trucks shows at one location instead of multiple locations. So we redesign the data generator in the simulator. Whenever the generator generates new data, it will include as many sets of data as the number of trucks. This will ensure that each car moves to a different location and takes a different route.

3.7 Results

None of testing have been performed yet.

4 Closing material

4.1 Conclusion

The goal of our project is to reduce time cost and financial costs and improve efficiency of snow plow operation combined with sensor system developed last semester.

Real time data from sensors employed in each truck will be took advantage to analyze decision of real-time reroute. MySQL is used to communicate and store data from AWS server which is provided by Henderson Products. Basic algorithm to meet company's goal will be implemented first and then more advanced algorithm may be discussed later. Main function of our projects will be achieved in two form: web and app. App will be built for snow driver to ensure that they understand condition of truck suck as supplies (sand, salt) and blade. Additional functions may be implemented in the future. Web-based interface can refresh and display the location of each truck and its historical route. And the date picker can be used to search historical data of all trucks for a period of time. It is mainly built for dispatcher who oversees overall snow operation and send assigned routes or reroutes information to drivers.

4.2 References

[1] *How Technology Is Transforming Snow Removal Operations*. EASTBANC Technology, *How Technology Is Transforming Snow Removal Operations*, www.eastbanctech.com/binaries/content/documents/site/technology-insights/what-the-tech/files/transforming-snowplow-fleet-operations/transforming-snowplow-fleet-operations/site%3Afile.

[2] *Using Data, Snow and Ice Control Fleet Managers Achieve Goals and Automate Key Systems*. Certifiedcirus.com, 2017, *Using Data, Snow and Ice Control Fleet Managers Achieve Goals and Automate Key Systems*, certifiedcirus.com/wp-content/uploads/2017/04/Certified-Cirus-Winter-Road-Maintenance-Data-and-Reports.pdf.

[3]"Fighting The Snow With A Fleet Management Solution (Case Study)." *GIS Cloud*, GIS Cloud, 21 Nov. 2018, www.giscloud.com/blog/fighting-the-snow-with-a-fleet-management-solution-a-case-study/.

[4]"Filling the Gap between Conception and Unit Test." *Software Engineering Stack Exchange*, StackExchange.com, softwareengineering.stackexchange.com/questions/276797/filling-the-gap-between-conception-and-unit-test.

[5]"HENDERSON PRODUCTS - ABOUT US." *About_Us*, Henderson Products, Inc., www.hendersonproducts.com/about_us.html.

[6]"Road Conditions, Speeds, Travel Times, Traffic Cameras, Live Streaming Traffic Cameras, Road Closures and Road Work Information Provided by Intelligent Transportation Systems (ITS) a Branch of Colorado Department of Transportation." *Road Conditions, Speeds, Travel Times, Traffic Cameras, Live Streaming Traffic Cameras, Road Closures and Road Work Information Provided by Intelligent Transportation Systems (ITS) a Branch of Colorado Department of Transportation*, COtrip.org, www.cotrip.org/snowplow.htm#/default?SnowplowId=7401.

[7]"Snow Removal Is Essential to Keep a Clear Path to Your Business." *IBHS*, IBHS, 1 Oct. 2018, disastersafety.org/ibhs/snow-removal-is-essential-to-keep-a-clear-path-to-your-business/.

[8]Talley, Becky. "Plow Tracker Site to Let Denver Monitor Snow Plow Locations." *Our Community Now at Colorado*, Our Community Now, 12 Oct. 2018,

colorado.ourcommunitynow.com/2018/10/12/plow-tracker-site-to-let-denver-monitor-snow-plow-locations/.

[9]TechMagic. "ReactJS vs Angular5 vs Vue.js - What to Choose in 2018?" *Medium.com*, Medium, 16 Mar. 2018, medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d.

[10]"What Is the Difference between a Distributed System and a Cloud Computing System?" *What Happens to the Planets When a Star Dies? - Quora*, www.quora.com/What-is-the-difference-between-a-distributed-system-and-a-cloud-computing-system.

[11]"Winter Maintenance of Roads and Runways." *Winter Maintenance RCM411*, Teconer, www.teconer.fi/en/winter.html.

[12]"Xamarin vs React Native vs Ionic vs NativeScript: Cross-Platform Mobile Frameworks Comparison." *AltexSoft*, AltexSoft.com, 4 Oct. 2018, www.altexsoft.com/blog/engineering/xamarin-vs-react-native-vs-ionic-vs-nativescript-cross-platform-mobile-frameworks-comparison/.

4.3 Appendices

No appendices available at this time.