Real Time Route Optimization

DESIGN DOCUMENT

Team Number: 18
Client: Adam Ryan
Adviser: Goce Trajcevski
Team Members/Roles:
Junjie Wen - Core Software/Algorithm Developer
Zhanghao Wen - Product Manager/APP Developer/Tester
Yuhang Xie – Backend Developer
Xinhe Yang – Frontend Developer
Tianhao Zhao – Communication Leader
Team Email: sdmay19-18@iastate.edu
Team Website: http://sdmay19-18.sd.ece.iastate.edu/

Revised: 10/17/2018/Version 1

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

Figure 1 Relation of designing tasks

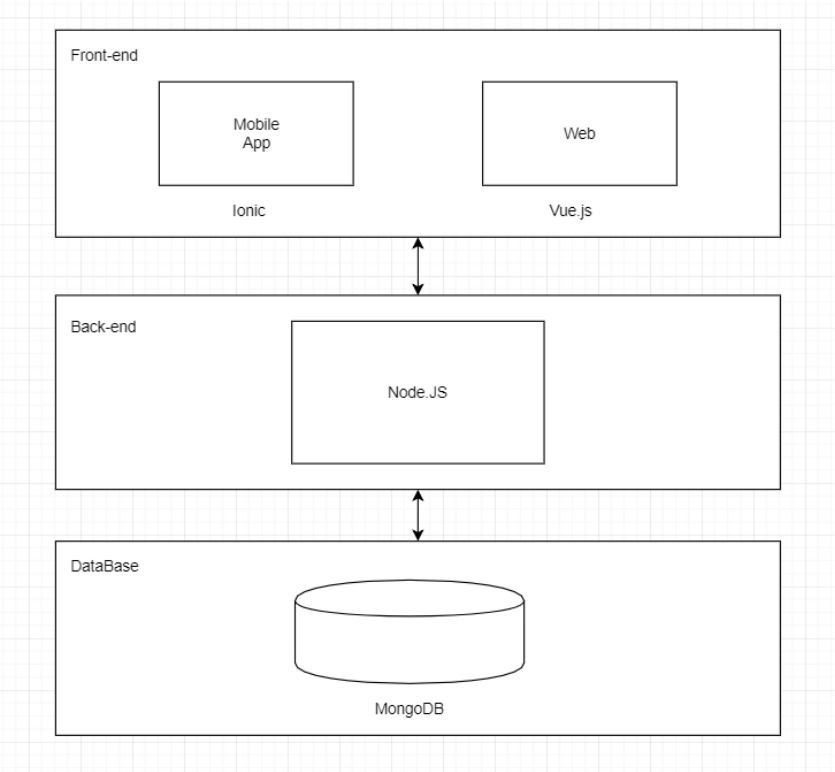
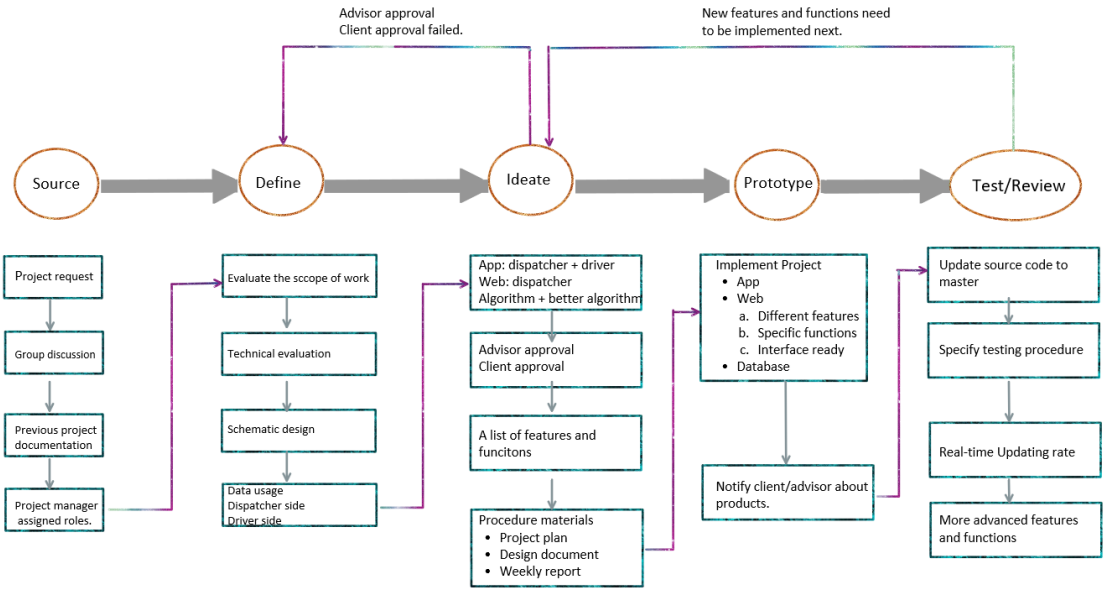


Figure 2 Designing process diagram



1. Introduction

1.1 Acknowledgement

Our client, Mr. Adam Ryan from Henderson Products Inc., will provide us the database runs out of sensors of machines for testing our solutions through progress, and pre-treat roadways for our initial route design. Our adviser, Prof. Goce Trajcevski, provides us with valuable advices on our decision and planning throughout the semester.

1.2 Problem and Project Statement

There is a lot of bad conditions of snow weathers in North America, which makes majority of inconvenience road conditions every year. There would be traffics in some situations if the snow plows have low outcomes or duplicated tasks in their operations. So, an efficient and manageable ice removal system is necessary.

In our solution, a web-based control system will be created to share necessary data between fleet of trucks and arrange the tasks efficiently. We will use snow plows' information that comes from sensors, such as supply details and locations. By sending to central computer with optimized solutions that runs out of our algorithm, the center control system will be able to manage and re-route the trucks for a better outcome.

1.3 Operational Environment

Our final product is a web application and will be used by dispatchers in Henderson Products Inc. Since it is a web-based design, it will not be relevant to physical environment. It will require to use specific local and online system with data that fits and operates the software solution from our design.

1.4 Intended Users and Uses

Our project Real-Time Route Optimization is to optimize the route of the snowplow and reduce the resources wasted during the movement. There are two types of user which is the Henderson's dispatcher and truck driver.

The dispatcher's task is to assign the route and purpose of each snowplow and return the base to refill the supply if the snowplow has few resources (local resources include: salt, Gasoline and sand, etc.). In this process, the snowplow may pass the same route and cause waste of resources. Our project is to help Henderson's dispatcher to properly distribute the snowplow and maximize resources.

Truck driver can use our mobile app to receive information from dispatcher and local sensor. This software can be used to display the location of the truck and receive instructions from the dispatcher. At the same time, the phone can display the local resources of its own truck, such as the salt and sand in the car, these resources can be transmitted to the phone through the sensors.

1.5 Assumptions and Limitations

Assumption:

1. We expect to complete the software by the end of March and conduct extensive testing in April to ensure that the final product will be available in May.
2. The maximum number of simultaneous users will not be limited.
3. Since the main colors on the company's icon are red and black, we will use these two colors on the most of interface.

4. The end-product will not be used outside the United States.

Limitations:

1. Because our final product is used to provide the best route for snowplow, our real-time testing must be carried out during the snowing period.
2. The amount of concurrent access to the software depends on the speed of the company's servers and the quality of the software's internal algorithms, so the final software cannot accept many concurrent accesses.

1.6 Expected End Product and Deliverables

In the end of this semester (Fall 2019), we are expected to get an efficient snowplow arrangement system (based on web), which can handle the data from sensor. Besides, the central computer can also show user the arrangement of snowplow. It will return several routers to user, which is the most efficient way to remove the snow. There should be least waste of snowplows' driving, which means the street that has been cleaned should not be clean again. In our final presentation, we should be able to show how to system works in a simulation environment.

2. Specifications and Analysis

2.1 Functional Requirements

- A web application, which must:
    - Display trucks' GPS location on map as well as traffic condition
    - Check out individual truck's supply condition (rate of water/salt dispenser)
    - Provide all the routes information which is ready to be assigned by dispatcher
    - Allow to communicate with truck driver
    - Allow to open weather condition on certain snow operation area
- A mobile application, which must:
    - Display trucks' GPS location on map
    - Check out individual truck's supply condition
    - Display company's name
    - Allow to communicate with operation center
- AWS server implemented by Nodejs, which must:
    - Generate new route information and send it to front-end
    - Authenticate driver/company to login
- MongoDB database, which contains:

2.2 Non-Functional requirements

- Modified truck's data in a database should be updated for all users accessing it within 1 minute.
- 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 Vue.js as our web design interface framework. It has a good html block handling since it is like Angular. 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 and Angular. 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 Vue.js.

2.4.2 Mobile application

Our mobile application is mainly targeted and designed for truck driver, so put different drivers may use different mobile operating system into considerations, we could build iOS app by using Objective C and Android app by using Java.

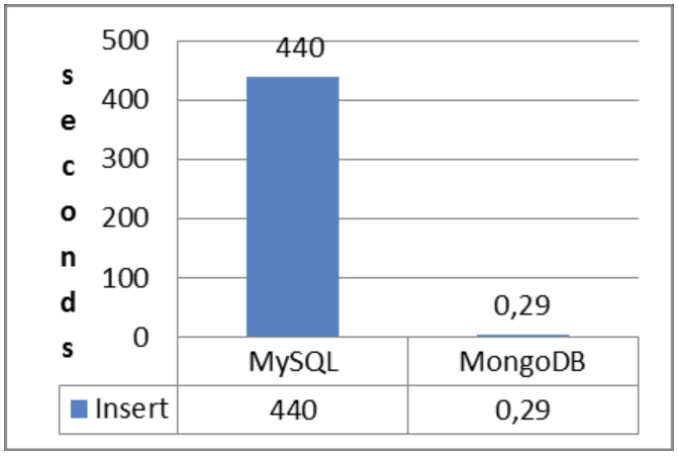
We could also build hybrid app. A hybrid app is a program that is built using HTML 5, CSS and JavaScript and wrapped in native container which can used in different platforms. We have choices like Xamarin, Ionic, Native Script framework.

2.4.3 DATABASE

Development: Mongo dB is friendly to developing engineer, because of its supply of JSON format data. With the Mongo DB, 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 Mongo DB 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, Mongo DB is better on loading.



2.4.4 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.

2.5 DESIGN ANALYSIS

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, o

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.

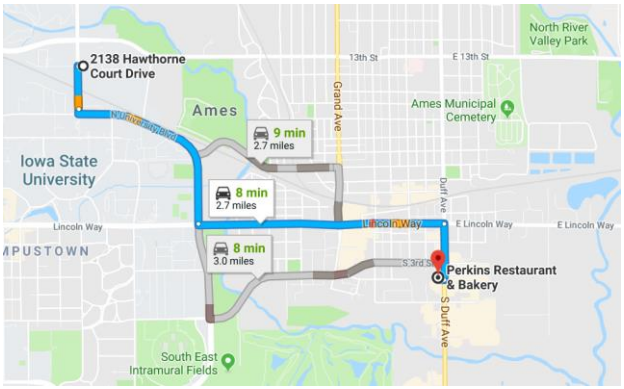
We chose NodeJs instead of Spring framework for the following reasons:

- 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 imporve development efficiency.

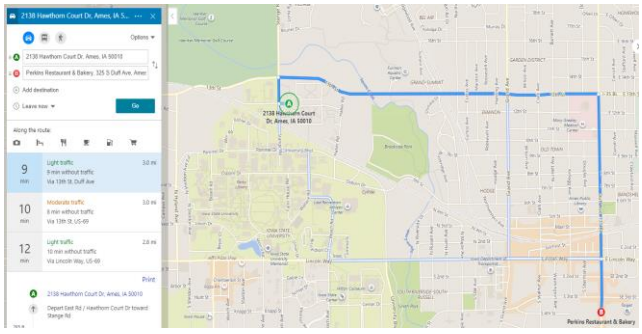
Spring is better than NodeJS in overall performance. However, we need more time, funds and people to develop a sophisticated project by using Spring. Under these premises, if we use spring as the back-end development framework, it will extend the time of our project delivery.

Google 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.



- 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.



- Leaflet: Adding a map to a simple website is definitely 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.

For display of the features and real time conditions, we create the frontend interface of both web and app for easier control and monitor during snow operation. There is a simple of tasks relations how we collect, use and display data in Figure 1.

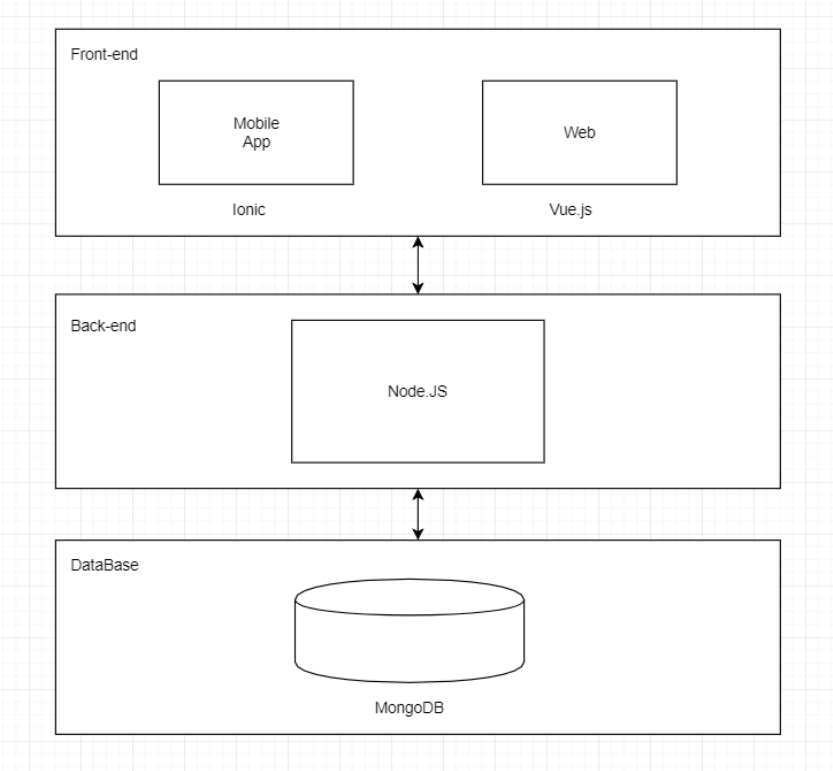


Figure 1

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.

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 nodejs to build our backend development framework. Finally we chose nodejs Here is an example of “Hello World” program by using Node.js and Spring:

Hello world in Node.js with express

```
const express = require('express' 4.16.4 )
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

Hello world in Spring

src/main/java/hello/Greeting.java

```
package hello;

public class Greeting {

    private final long id;
    private final String content;

    public Greeting(long id, String content) {
        this.id = id;
        this.content = content;
    }

    public long getId() {
        return id;
    }

    public String getContent() {
        return content;
    }
}
```

src/main/java/hello/GreetingController.java

```
package hello;

import java.util.concurrent.atomic.AtomicLong;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class GreetingController {

    private static final String template = "Hello, %s!";
    private final AtomicLong counter = new AtomicLong();

    @RequestMapping("/greeting")
    public Greeting greeting(@RequestParam(value="name", defaultValue="World")
        return new Greeting(counter.incrementAndGet(),
                            String.format(template, name));
    }
}
```

```java
package hello;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Clearly, we can see that the amount of code difference between two framework.

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.

3.3 Functional Testing

Unit test: A bunch of unit tests should be built, which will cover all platforms, including android, IOS, front-end and back end code. And should be executed every time the code be changed.

Integration test: There should be individual integration test for different platform, including Android, IOS, Front-end and Backend.

System test: Test engineers should implement test code based on black box test policy.

Acceptance testing: The whole team will work on final evaluation of acceptance.

3.4 Non-Functional Testing

Performance testing: Backend platform performance testing should be built based on JMeter. Front-end code performance test will build based on status.

Security test: security test will be implemented based on black box test policy.

Usability: All team member should participate in usability evaluation.

Compatibility: For compatibility testing, we will do compatibility test with virtual machine (for different version of mobile platform) and multiple version of browse.

3.5 Process

Our planning of tasks shows as Figure2 below.

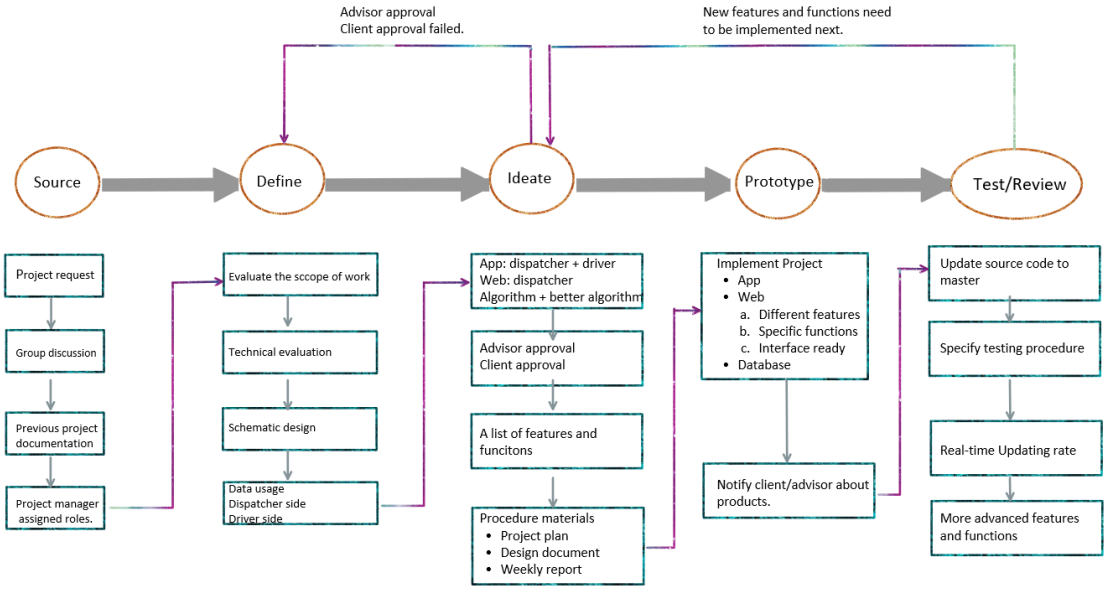


Figure 2

3.6 Results

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 will be built for dispatcher who oversees overall snow operation and send assigned routes or reroutes information to drivers.

4.2 References

https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d

https://www.altexsoft.com/blog/engineering/xamarin-vs-react-native-vs-ionic-vs-nativescript-cross-platform-mobile-frameworks-comparison/

4.3 Appendices