Team No. 2
Grant Gollier, Tiernon Riesenmy, Antonette Gichohu, John Quitno, Michael Svoren

Project Proposal

Project Name: *CarFlo*

Project Synopsis: Dynamic traffic management for cities

Project Description:

Sitting in traffic is frustrating. The goal of *CarFlo* is to minimize the time spent in traffic by creating a dynamic traffic management system. Traffic is a consequence of the increased density of cars on a road. Intuitively, the faster vehicles move, the longer the distance required between each car. Thus, as density increases, vehicles naturally start to slow down to maintain a safe distance. This process occurs naturally in a choppy manner, with inconsistent acceleration and deceleration, leading to congestion. If speed limits could change dynamically to reflect the quantity of vehicles on a given section of highway, hypothetically some of this congestion could be alleviated. As autonomous vehicles take the roads and have their own lanes, they could handle dynamic speed limits with ease. This presents an opportunity to build for this problem.

CarFlo is a traffic management solution which suggests dynamic speed limit modifications as a consequence of vehicle density and weather on a given roadway. Using computer vision, *CarFlo* interprets roadway capacity levels and uses a model to predict the optimal speed limit required to reduce congestion while maintaining traffic flow. The predictions are sent from the service to electronic speed limit signs. Additionally, there is a user-interface for managing associated settings. This project will serve as a proof of concept of the CarFlo hypothesis.

Project Milestones:

First Semester:

- Design mockups and user workflows (October 12)
- Proposal Video (October 26)
- Produce document detailing standard traffic flow modeling (Partial October 26, Full November 3)
- Generate software implementation plan document (November 24)

Second Semester:

- Generate simulated input (February 19)
- Back-end implementation (March 12)
- Implement user interface (March 26)
- Traffic flow visual simulation - Stretch goal (April 23)
- Project Poster - (May 1)

Project Budget:

Item	**Price**	**Vendor**	**Date Needed**
Photogate	$11.50	Digi-Key	03/03/2021
AWS Credits	$50	Amazon Web Services	Monthly (starting 02/01/2021)
Filmora (Video Editing)	$40	Filmora	10/24/2020

Work Plan:

Grant:
- Backend implementation and design
- Traffic Flow Visualization

Tiernon:
- Machine Learning Implementation
- Design Mock-ups and UI

Antonette:
- Machine Learning Implementation
- User Interface

John:
- Simulate traffic flow input
- Traffic Flow Modeling

Michael:
- Backend implementation and design
- Traffic Flow Modeling

Project Design

Overview:

CarFlo is a traffic management system that decreases congestion by dynamically varying roadway speed limits based on traffic density, weather conditions, and road impediments such as accidents. CarFlo will interpret these data points in real-time to recommend a speed limit that optimizes traffic flow.

Carflo's target customers include city and state authorities. With a trillion dollar federal transportation bill on the near horizon and the emerging ubiquity of autonomous vehicles over the next 10 years, now is the time to innovate in the transportation sector.

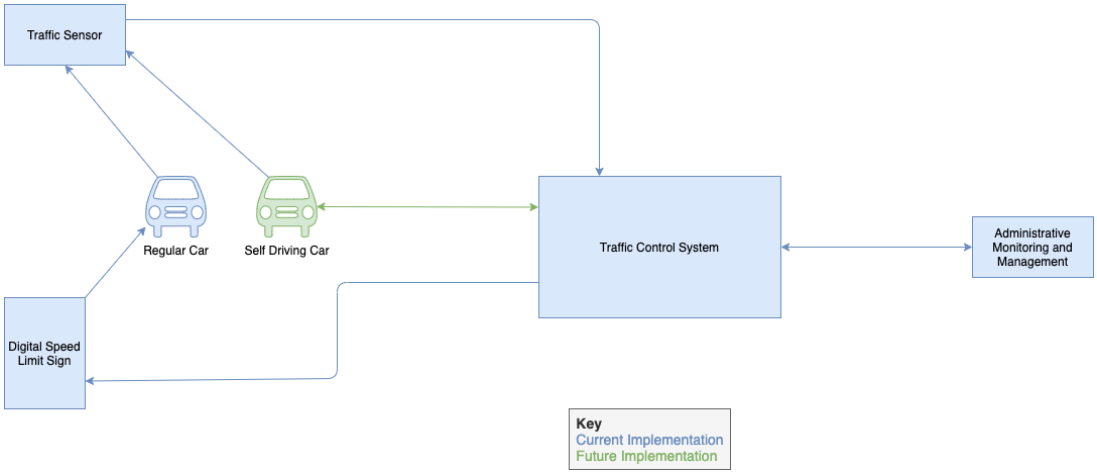
The product will have three core components. First is the traffic model. This is as discussed in the first paragraph, but will read in data from sensors deployed on the freeway related to traffic density, weather, etc., then recommend an optimal speed limit. This is a cloud-based platform, so all data is handled and interpreted on the backend systems.

The second component is the dynamic speed limit output. The output from the aforementioned traffic model is broadcast to the dynamic speed limit signs on the road, where they are updated. These signs exist currently, but are not common. Additionally, there will be an API for integration with outside services (e.g. Google Maps) to pull the speed limit for a given road at any time. This is particularly important as Autonomous Vehicles deploy with increasing frequency. Providing a live data source for critical information will increase the reliability of the AVs by supplementing their dependency on computer vision for sign recognition.

The third component of the product is the user interface. As stated earlier, the target customers of CarFlo are transportation authorities. These figures will need the following key features to manage the traffic system in their jurisdiction:

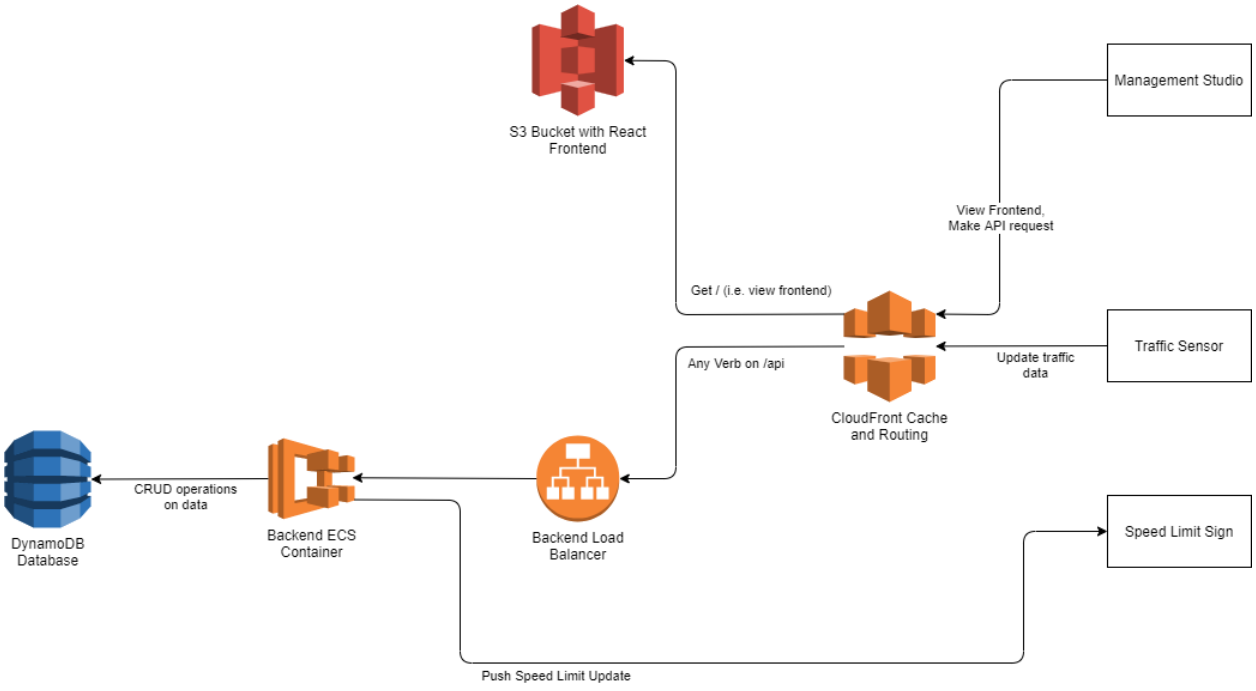
- Add speed limit signs to a specific location
- Remove speed limit signs
- Edit sign details
- Real-time view of live speed limits
- Override/Modify Speed limits
    - Temporarily disable
    - Manual modification
- Add/Remove constraints to automated speed limits
    - Time constraints
    - Maximum/minimum

With the exception of editing sign details, these features comprise the minimum viable product. Transportation authorities will have a web-based platform where they may control their deployment of CarFlo, using the features listed above. A diagram showing the flow of information is attached below.



Tech Stack

For our tech stack, we want to balance both functionality and scalability with ease of use. For us this meant finding the right level of abstraction that allowed us to focus on the technology and our application without diving into managing virtual machines and operating system updates. However, for ease of use we wanted something that would allow us to develop locally. Though self-imposed, this constraint is important since things like AWS Lambda require you to upload your code whenever you want to test something which slows down development speed. Additionally, we wanted something that we were already fairly comfortable with so we would not have to spend a lot of time getting up to speed with a bunch of new technologies.



With all of this in mind, we came up with the above tech stack which is a pretty standard single page application frontend with a monolithic backend. Specifically, we are using React for

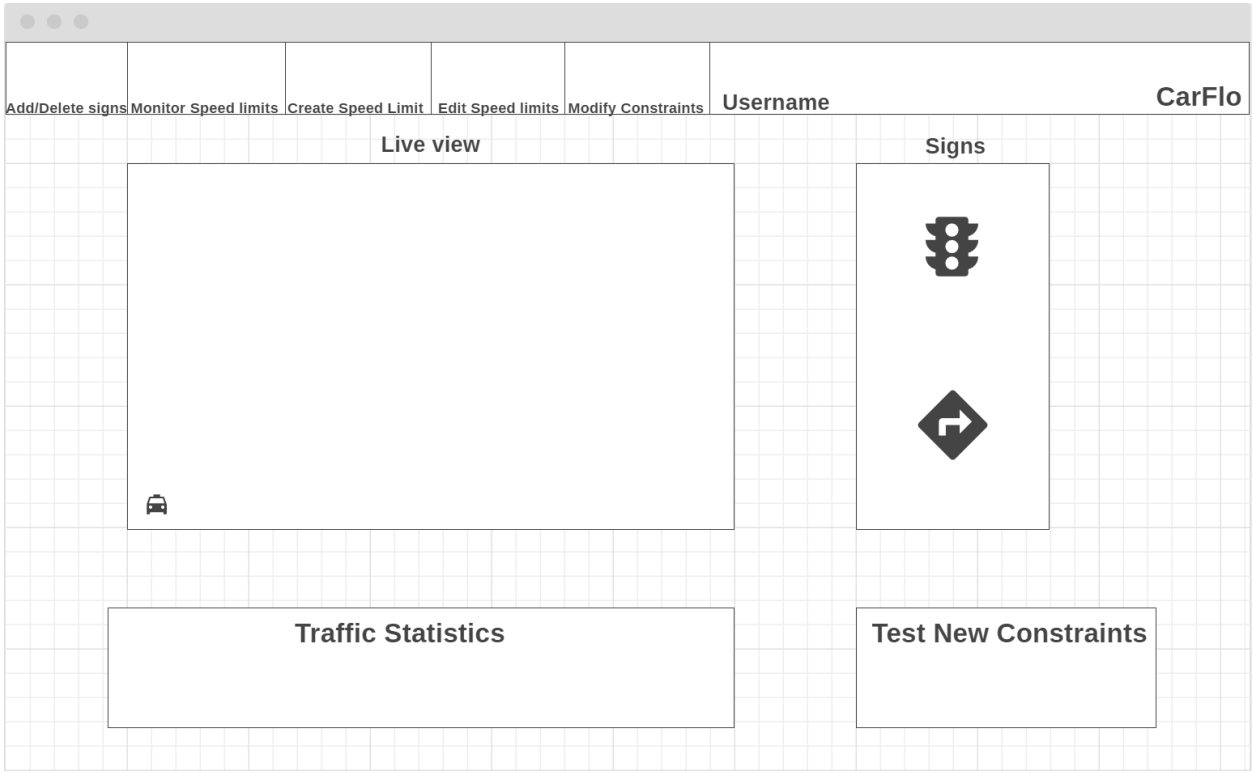
our frontend which we will host on S3 with CloudFront in front as a cache and routing layer. Then on the other side we will have our backend also behind CloudFront behind the '/api' route. This will be routed to an Application Load Balancer which will route requests to our containerized backend hosted on AWS ECS backed by Fargate. The Application Load Balancer serves two purposes. First, it is the only way to connect CloudFront to an application in ECS. Second, and more importantly, it allows us to run multiple instances of our backend which lets us scale horizontally as load increases. We are using DynamoDB as our database to back our application due to its low cost and ease of use compared to an RDMS like RDS. Finally, we are planning on having the traffic sensors push data into our application and then when the model decides that a speed limit needs to be changed it will push that change out to the necessary speed limit signs.

Model

Our model is going to be a python simulation that allows us to test configurable parameters and traffic patterns. Our project goal is to be able to adjust speed limits and other parameters to improve the flow of traffic. The first thing that we did was make sure that this is an effective way of improving the throughput of traffic. There are several academic papers that go over implementations of this in the real world. It isn't popularized and hasn't been done on a large scale due to logistical concerns, but it is the case that traffic is improved when you correctly adjust these parameters. Many of the implementations of this in the real world use a lot of complex math and statistical analysis to reach conclusions about the optimal configuration. We don't necessarily have the skillset or time to go as in depth as some of these papers, which is a distinct constraint, but our approach is simple enough to understand, and will motivate the use/effectiveness of being able to control speed limits. The first mathematical model that we looked at is the Greenshields model. It is this simple law that states that Speed = A – B * Density. Where A is the free flow constant and B is some observed constant. This is just saying that speed decreases linearly with the increase in density of cars on the road. Looking at throughput, Speed*Density, and substituting this equation in we get that Throughput = A*Density – B * Density^2. This is basically saying that the throughput of cars is an upside-down parabola as a function of density. So, when density is 0 and there are no cars on the road and the throughput is 0, and when the density is really high, the cars are locked in a traffic jam and the throughput is again 0. So, there is some optimal speed of cars to maximize throughput here. This becomes much more difficult when you add in traffic complexities. For example, when it is rush hour and you have 2 separate highways that merge, what should the respective speed limits be to maximize expected throughput of the merged highway? These questions become too hard to answer using the simple model when you introduce these things. This is where our python model will come into play. We will have a grid that represents some traffic configuration, and then simulate the movement of cars with the ability to change speed limit parameters. This will give us the ability to find near optimal parameters to maximize the throughput of these cars using some iterative guess and check like methods.

Front-End

We will implement the front end using React as it is the web framework with which we are most familiar. The goal with our UX design is to implement an easy to use interface that is also minimal. We will have a login page that asks the user to input login credentials. After authentication, the user will be redirected to their dashboard. The dashboard will consist of a live traffic view with traffic statistics and access to our tool suite. To maintain usability while dealing with the constraint of screen space, we must ensure that the dashboard is not cluttered and by default displays only the most relevant information. Additionally, we must list each individual tool from the tool suite in a clear manner so that users understand which command to use without having to frequently refer to a manual. The following figure is a wireframe outlining the appearance of the dashboard,



The information from our model must also be interpreted in a comprehensive but understandable way. When a user runs a test using new constraints, the website must clearly communicate the result. Since React itself does handle graphics visualization, we may simply provide the user with numerical values to indicate the results of a test. Further, we will have to employ a third party tool to handle the live traffic feed and overlay our own information. Otherwise, the UX design will remain minimal to provide a simple experience and work within the limitations of React.

Ethical Issues

Public Infrastructure:

A major ethical concern of CarFlo is its deep integration with public infrastructure. Since CarFlo is directly controlling speed limit signs on public roads, it needs to be both safe and reliable. This is especially important in terms of uptime during upgrades as people will be driving on highways at all times of the day. As discussed in the ACM Code of Ethics section 3.7, we must be extremely careful with how we roll out changes since CarFlo is by design an integral part of public infrastructure. Moreover, when cities want to leave CarFlo we need to make sure there is a safe way for them to offboard with their speed limit signs continuing to work until they have completely migrated to a different system. Finally, as discussed in that same section of the ACM Code of Ethics, we need to seek out and help create where necessary, rules and guidelines for the safe operation of public infrastructure such as this to ensure not only the safe operation of our systems, but also the reliable operation of that of our competitors. This is so that as an industry of automated public infrastructure, we can ensure that everyone is operating safely.

Choosing what's safe or choosing what customer requests (if they contradict):

A second ethical concern with CarFlo relates to the critical decisions the product makes. Specifically, there could be a situation in which a city or municipality makes a request to modify the speed limit or model parameters in a way that is shown to be less safe than the optimal way. For example, the authorities could face local political pressure to set the speed limit maximum constraint at 60mph for a given set of conditions on a given length of road. However, CarFlo may determine that the maximum speed should actually be 55mph. In this case, a decision would have to be made whether to risk the sale and insist on the lower maximum speed or make the sale and go above this determined safe limit. The ethical solution to this problem would be to not compromise on safety and simply accept the possibility that a sale may be lost. However, an added benefit would be that this action could build trust in the brand over a long period of time.

Intellectual Property Issues

Adapting someone else's model:

There are a couple ways that we will be considering intellectual property in relation to our model. We will reference some research papers to build and think about the mathematical models that govern traffic. We will also reference some implementations of traffic simulations in python. Our goal is not to directly copy these things, but use their work as a reference to build our software. We will consider the licenses under which this software was developed, however, most of the projects that we have seen so far were written in some academic context. We don't plan on making money through this project so we should have no problem using open source software as a reference.

Using 3rd party software libraries:

Most of the risk with using third party software libraries would come from not properly crediting the author of the library, or using the software library in a way it was not intended to be used. We plan on keeping close track of how we utilize these libraries to make sure we are not infringing on the author's vision of how their work is to be used. Another issue could stem from monetizing the product without checking on the requirements for doing so, but since we do not plan to make money from this product, this shouldn't be an issue.

Change Log

- Budget: We added Vendor and Date to the budget table. Also, we clarified our sensor requirement to specify vendor and sensor type to emphasize transparency in our purchasing process. Finally, we added a video editing tool to create our pitch proposal video.
- Project Description: We made note of the decision to incorporate weather from the OpenWeather API into our model, which was not included in the original proposal.
- Work plan: Antonette and John had roles adjusted to respond to our more developed priorities.

Gantt Chart:

The Gantt Chart is attached on the next page.

CarFlo Gantt Chart

PROJECT TITLE	CarFlo	**COMPANY NAME**	University of Kansas
PROJECT MANAGER	David Orville Johnson	**DATE**	10/1/20

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**First Semester**					
1.1	Design mockups and user workflows	John Q. /Tiernon R.	10/1/20	10/12/20	11	100%
1.1.1	Proposal Video	Antonette G.	10/2/20	10/26/20	24	100%
1.2	Research & produce document detailing standard traffic flow modeling	Michael S. / John Q.	10/3/20	11/3/20	30	50%
1.3	Generate Software implementation design document	Grant G.	10/1/20	11/24/20	53	35%
2	**Second Semester**					
2.1	Generate simulated traffic input	John Q.	2/1/21	2/19/21	18	0%
2.2	Backend implemantation	Grant G./ Michael S.	2/4/21	3/12/21	36	0%
2.3	Implement user interface	Antonette G/Tiernon R.	2/2/21	3/26/21	52	0%
2.4	Traffic flow visual simulation	Tiernon R.	2/3/21	04/23/21	60	0%
2.5	Project Poster	Everyone	4/24/21	5/1/21	20	0%