DUDETRUCK

Aaron Hartigan	219247094
Matthew Nix	217474596
Thomas Hoang	219484409
Alexandru Seremet	219247393
Christel Garcia	219563358
Lincoln Gallegos	215590766
John Veit	218751027
Vahak Gilian	215258590

System Architecture

Components:

*****route specifies a component that leads to its own path in the URL.

< Home /> (route)

Contains a welcome message and links to <Login/> and <Register/>

< Login /> (route)

Contains fields to enter email and password. On successful login, redirects to <Search/>. On a failed login, redirects to < Login/> and displays an appropriate error message.

< Register /> (route)

Contains fields to enter email, password, confirm password, and what type of user they are (food truck vendor or non-vendor). On successful registration, redirects user to <Login/>. On failed registration, redirects to <Register/> and displays an appropriate error message.

< Search /> (route)

Automatically asks the user for permission to location data. If rejected, the app will still work, but it will not automatically follow the user, and the map will not be automatically zoomed in. If permission to location data is accepted, the map will zoom in the user's location and display any nearby food trucks. The food trucks will automatically be filtered by the user's preferences. A listing of all nearby food trucks will also display beneath the map. The listings will contain the food truck's picture, rating, and a short description. The listings will be clickable and redirect to <Vendor/> and have complete information on the food truck.

< Settings /> (route)

Depending if the user type, this page will display settings a user can change. If the user is a vendor, they can edit their picture, description, location, and food types available. A non-vendor user will be able to edit their search preferences (vegan, gluten-free, etc.).

< Vendor /> (route)

Accessible from the < Search /> component. Receives a vendorID to display the appropriate vendor. Displays the vendor's picture, description, food types available, rating, location, and all reviews. Also displays < Reviews />.

< Reviews />

Receives a vendorId to fetch the appropriate vendor's reviews and displays them.

Database Schema:

User:

id	UUID	primaryKey
email	STRING	
password	STRING	(hashed with bcrypt)
type	STRING	(vendor | user)
isVegetarian	BOOL	
isVegan	BOOL	
isGlutenFree	BOOL	

User.hasMany(Review)

Vendor:

id	UUID	primaryKey
logo	STRING	(link to AWS S3)

lat	FLOAT
log	FLOAT
description	STRING
isVegetarian	BOOL
isVegan	BOOL
isGlutenFree	BOOL

Vendor.hasMany(Review)

Review:

id	INT	primaryKey
vendorId	UUID	foreignKey
userId	UUID	foreignKey
text	STRING	
rating	INT	(1-5)

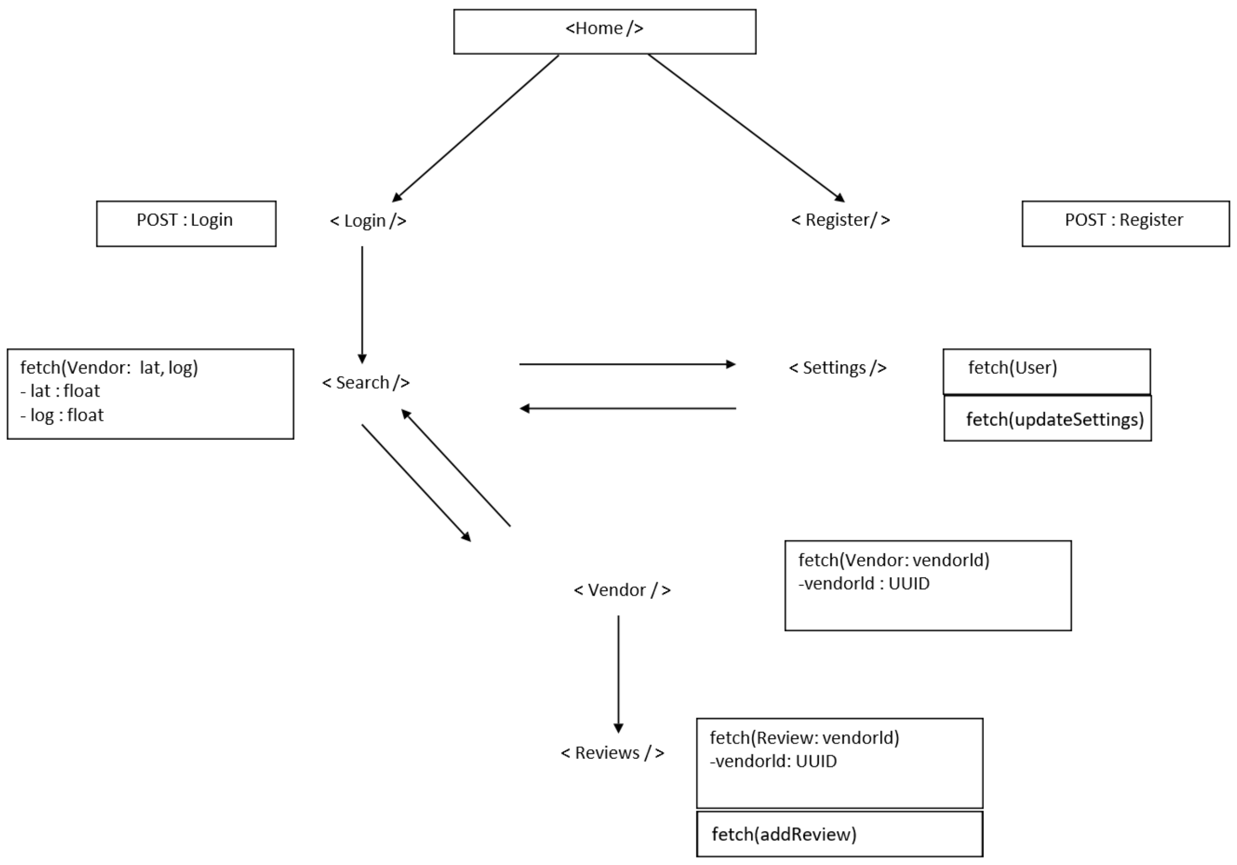
Design Alternatives:

In the database, we had the option of making a separate table for UserSettings. We chose to incorporate UserSettings directly into the User table. This will make database lookups slightly easier, with the downside being that if we ever add more settings in the future, database migrations become more complicated.

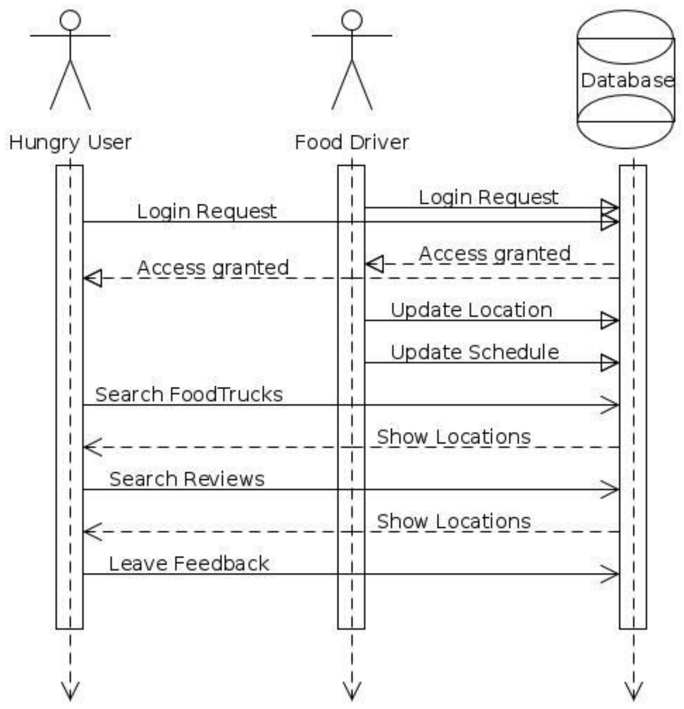
We also had the option of using React Native. React Native is certainly more natural for creating a mobile web app. However, our group had more experience with React than with React Native. In addition, better tools exist for creating React apps. And finally, it is easier to test a web app (with a web browser) than a native app (using some sort of emulator). Since our project has a strict deadline of one semester, we decided to use the tool we thought we would be able to accomplish the most with, which is React.

Diagrams:

"UML" diagram:



Use case diagram:



Process

Risk Assessment:

Not being able to locate the food trucks

- Likelihood: medium
- Impact: High, main focus of the app
- Evidence: Google Maps API may not be able to track the user accurately or display the locations of the food trucks in the first place
- Steps: Focusing on the backend, constantly trying to break the app when it comes to gathering user location data as well as food truck location data

- Plan: Use Trello to track any bugs, as well as utilize one of our group members to try and break the app on a constant basis so that we can determine what bugs there are and how they can be fixed
- This risk has not changed much since the SRS as it is the main focus of the app and it is something we are constantly on the lookout for

Not being able to filter user search results based on preferences

- Likelihood: low
- Impact: medium, as it is important to the app, but not the main focus
- Evidence: None presented so far, as we have not created user profiles just yet, but is something we are on the lookout for
- Steps: If problems arise, we would put more focus onto that portion until we figure it out as it is still a big part of the app
- Plan: Use Trello to track any bugs, as well as utilize one of our group members to try and break the app on a constant basis so that we can determine what bugs there are and how they can be fixed
- This risk has not changed since the SRS as we are still getting to that part of the app, but it is still a main focus for when we get to it

Project Schedule:

DD = Developer Day

Familiarize self with our tools: 1 DD per developer

Implement Routes and Components Skeleton Code: 1 DD

- This must happen first. After this, any component can be worked on in any order.

- The backend will be handled with GraphQL, and it does not need to be implemented for the frontend to work.

Implement Getting User Location: 1 DD

Incorporate Google Maps: 3 DD

Implement Login/Register: 1 DD

Implement GraphQL Types and Skeleton Code (no resolve functions): 3 DD

Implement GraphQL resolve functions: 2 DD per GraphQL resolve

- Currently we have 7 GraphQL queries planned

Implement Frontend Barebones: 1 DD per component

Implement Frontend Aesthetics: 1 DD per component.

Total: ~45 DD.

Team Structure:

project manager for DudeTruck is Matthew Nix. We have selected to separate the roles out at this juncture. The UI designers are Matthew Nix, and Vahak Gillian. They have a vision of how the end product will look. The frontend development team contains Aaron Hartigan, Alexandru Seremet, and Lincoln Gallegos. They volunteered to do this part of the project. The backend team contains Thomas Hoang and Christel Garcia. They have experience with background servers and their communication. The tester will be John Veit. He likes looking for ways to improve a product through the creation process. As the project progresses these roles may change as one portion of the project may require more or less man power. We shall handle disagreements as adults by talking them out and coming to a mutual agreement on the subject. We believe this structure shall allow our group the best chance at success for now and in the future.

Test Plan:

John Viet will be in charge of manually testing the app. In addition, the developers will be creating automated tests using Mocha and Jest. These tests will be run automatically before each commit (code will not be able to be committed without passing the tests). Any bugs found will be documented with Trello. Any bugs fixed will have a test case written so they do not occur again.

Documentation Plan:

Constantly updating the files online, as well as keeping track of chat logs and conversations that we have through the group Discord channel. Discord, the better way to chat and do projects.

Coding Style Guidelines:

We are following AirBnb's JavaScript style guide -
https://github.com/airbnb/javascript/blob/master/README.md

In addition, our code is run against a linter configured to this style guide. No code will be able to be committed without passing the linter.