Requirements Specifications

Authors:	Meng Cha, Taylor Meyer, Brandon Nhem, Chris Tran
Group Name:	byteMe
Date:	September 19, 2019
Version:	1.0

Table of Contents

Executive Summary

7!0 (working title) is an application that would enable users to quickly, and with no hassle, to split a gas bill with other passengers and/or the driver.

Currently, you would need to pay with cash on hand, find an ATM, or use an application that requires your bank account information. Our application aims to be a tool to quickly determine a fuel cost all passengers can spend and give support to various pay service APIs.

In this document we will detail who the stakeholders are in this application. Then, indicate the basic and future functionality of what the application needs and how it will grow.

We will diagram and document the overall system vision for the application and our goal model.

Lastly, we will diagram the overall usage model and its individual use cases that will be needed to properly implement the basic functionality.

Other than create a useful app, we hope to have an impact on stopping climate change. We want to be able to quantify the positive change people make by choosing to carpool instead of every individual traveling in their own vehicle.

Introduction

Problem Analysis

If you want to split a gas bill with a friend, currently you would need to already have the cash with you, stop at the ATM, or use an application that requires you to hand over your bank account or card numbers. It's not always the case your carpool is using the same bank as you.

If you are not able to do any of those things, you might get backed up into a cycle of reminding each other, offering to pay the next time you go out, and so on.

Climate change has found its way into every aspect of our lives and a big factor is the vehicles we drive and the amount of vehicles on the road. Commuting around in a sporty car in Southern California is not the best thing for the environment or your wallet. Not to mention the traffic and parking congestion that everyone in Los Angeles must deal with on a regular basis.

Think about the parking at CSULB for instance. The school just took in 11,000 students in the Fall 2019 semester. Virtually none of them are living in a dorm, they are all commuting to school one way or another.

Every semester the amount of vehicles on campus goes up without any solutions to alleviate parking.

Solution

Instead of finding an ATM, or the carpool downloading a "cash" app that they will ever use once, our application saves the day. Using google maps, we would take your current location and your destination. Using the miles of that trip, multiplied with an average price of fuel, divided by the number of people splitting the bill, we will show you what your contribution to the trip will be.

Then you would be able to choose any number of pay services (starting with PayPal) and use their API through our application to pay the driver. If you do want to pay with the cash you have our hand, you can use our app to get the cost and then close out. Currently, there is no need for a user to give us any personal or financial information to complete the process.

We also wish to show the user a quantifiable view of how carpooling is helping climate change. Stopping climate change is a big issue and most are not willing to do anything unless the problem gets fixed all at once. But that is not realistic, and we will try and make a small impact.

By showing how carpooling is helping by better cars off the road, hopefully we users will feel encouraged to continue to do so.

A side effect of this would be alleviating the aforementioned parking issues if we stop every individual from using their own car when they go out with their friends.

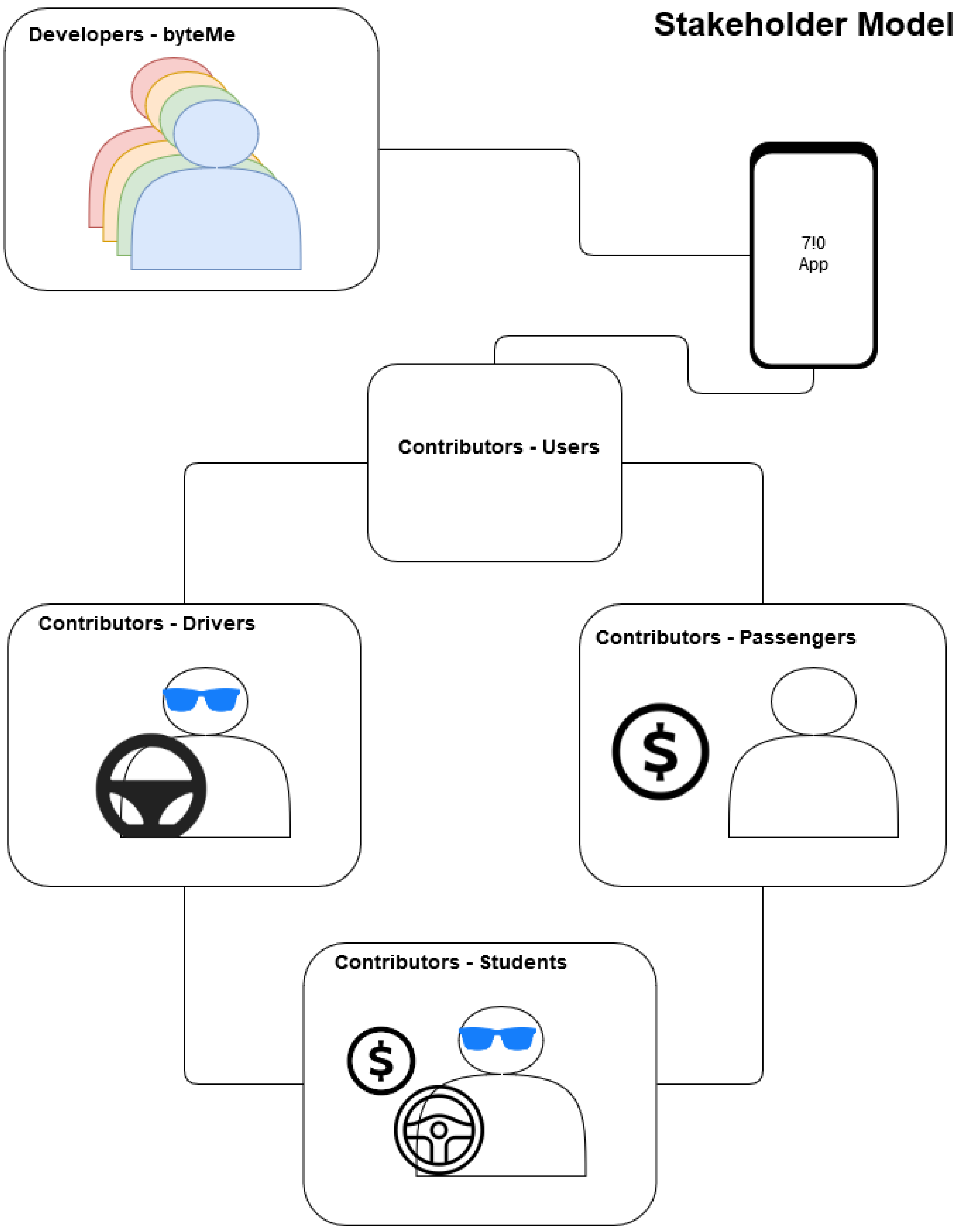
Stakeholder Model

A stakeholder is an entity with an interest or concern in something. We provide a detail for how each entity belongs in a category which serves as a role for our system.

Our stakeholders involve the developers and contributors. The contributors are what makes the app work as they are vital to our success. Ultimately there is only one type of contributor which is a users which is broken down into two subtypes, drivers or passengers.

Drivers are the ones to get paid for driving whereas the passengers are the users paying the drivers.

Once more, a subcategory is made to include students which we would like to address as they could help contribute to lower emissions.


Stakeholder Model
Developers - byteMe
7!0
App
Contributors - Users
Contributors - Drivers
Contributors - Passengers
Contributors - Students

Stakeholders

Developers

Representatives	Meng Cha, Taylor Meyer, Brandon Nhem, Chris Tran
Description	Developers of Software
Type(s)	Software Engineer
Responsibilities	Provide software for all users
Success Criteria	Creating software that is simple and intuitive for all users
Involvement	Handling development and upkeep of software
Deliverables	Source code, requirements, working application

Users

Representatives	Drivers, Passengers
Description	Targeted demographic of users that have an aim to rideshare whether that means driving or sitting with the driver
Responsibilities	Providing destinations and paying rides
Success Criteria	Using the application properly without any need for assistance
Involvement	Offering rides or paying for rides
Deliverables	Destination Address, Payment Services, Email, and Phone Number

Goal Model

A goal is an aim or desired result. We have three goals that we would like to achieve.

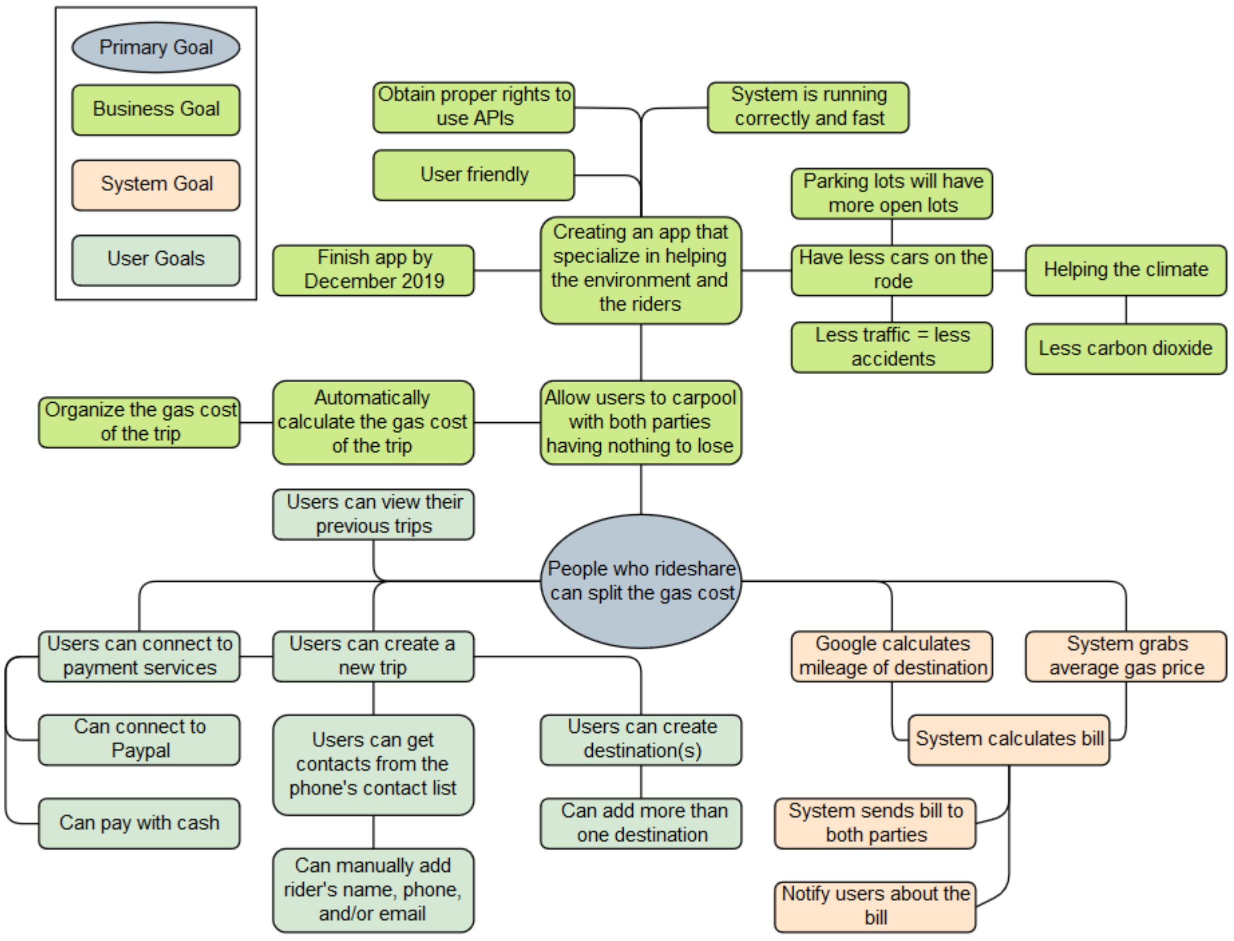
Our goals are a user-friendly application, a parking goal, and an environmental goal.

It goes without saying, applications that are too much of a pain to use quickly fade and never pick up any traction. While we are not looking to solve a major world issue, we will consider this application a success when it becomes an application we ourselves would use.

Next, our parking goal is to decrease the amount of students waiting for parking by encouraging ridesharing. This is quantifiable by measuring how long it takes for students to find parking.

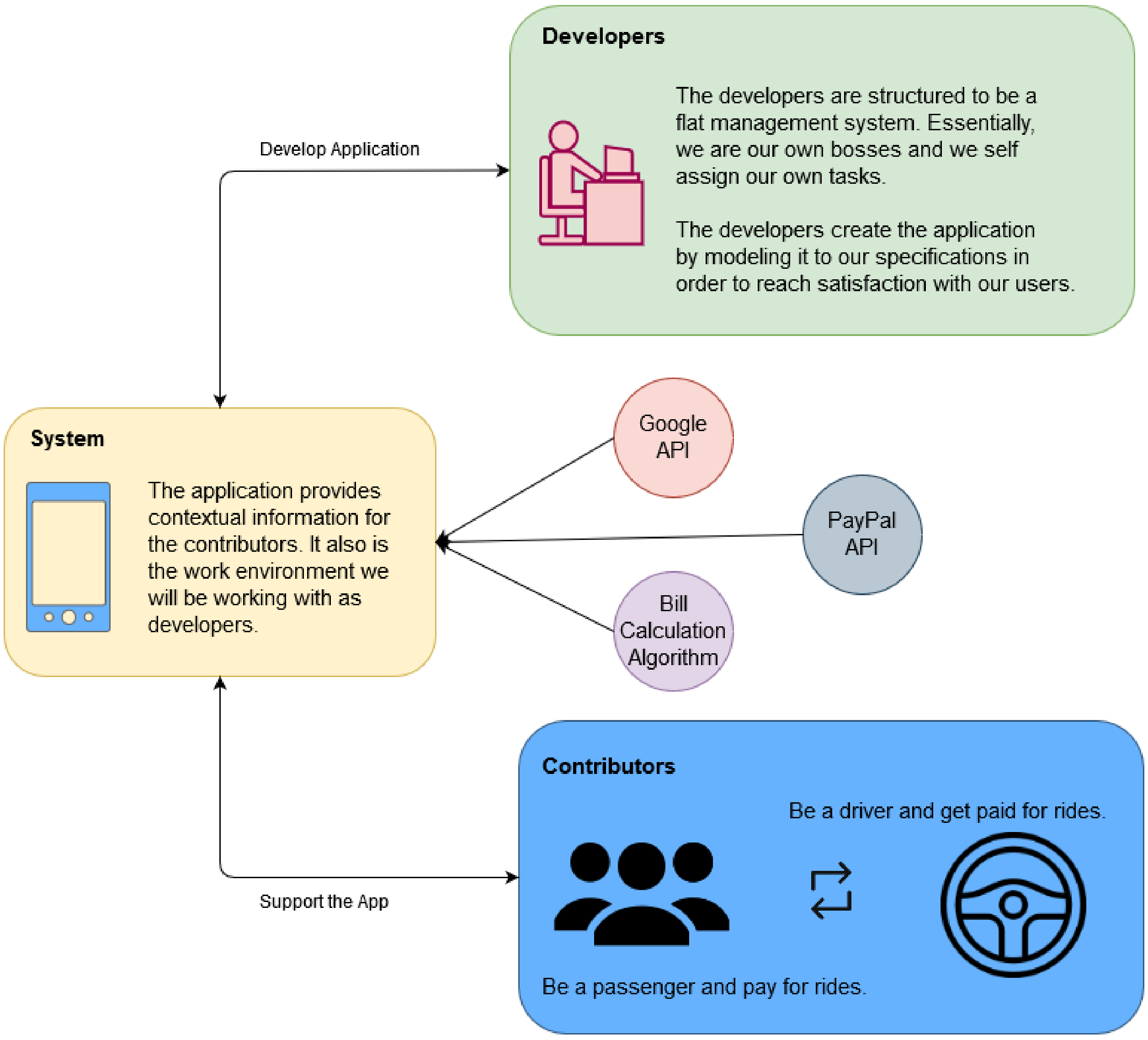
Lastly, our environmental goal is to significantly reduce carbon emissions by encouraging many to rideshare rather than driving alone. With more drivers filling up their cars, we would have less drivers on the road creating carbon emissions. We intend to find a way to quantify this for the users.

Goal Model Diagram


Primary Goal
Business Goal
System Goal
User Goals
Obtain proper rights to use APIs
System is running correctly and fast
User friendly
Parking lots will have more open lots
Finish app by December 2019
Creating an app that specialize in helping the environment and the riders
Have less cars on the rode
Helping the climate
Less traffic = less accidents
Less carbon dioxide
Organize the gas cost of the trip
Automatically calculate the gas cost of the trip
Allow users to carpool with both parties having nothing to lose
Users can view their previous trips
People who rideshare can split the gas cost
Users can connect to payment services
Users can create a new trip
Google calculates mileage of destination
System grabs average gas price
Can connect to Paypal
Users can get contacts from the phone's contact list
Users can create destination(s)
System calculates bill
Can pay with cash
Can add more than one destination
System sends bill to both parties
Can manually add rider's name, phone, and/or email
Notify users about the bill

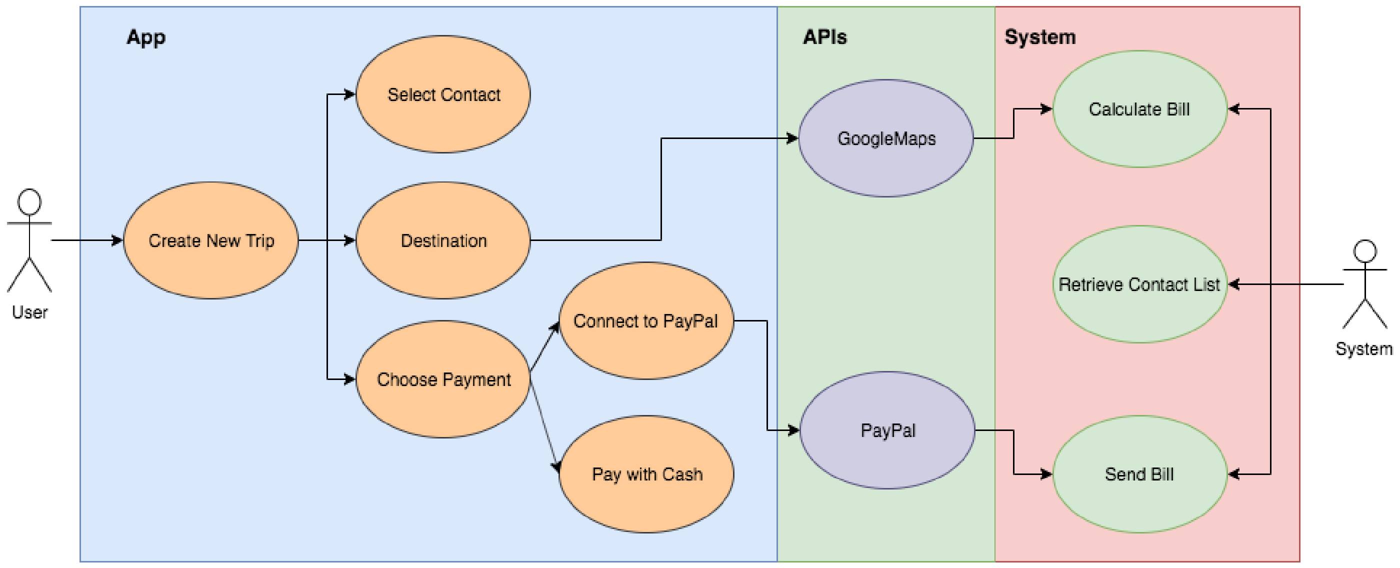
System Vision

Our system vision is the relationship between multiple entities such as our system, our contributors, and our developers. The system vision is depicted below which showcases the relationship between the entities mentioned before. Each entity affects the other in its own way by influencing where the direction of the project goes. To start off, the developers are responsible for maintaining and updating the system. The contributors are responsible for managing the payments for either the driver or the passenger. Lastly, the system itself unites the two, the developers and contributors, in which it provides the service on both ends.


Developers
The developers are structured to be a flat management system. Essentially, we are our own bosses and we self assign our own tasks.
The developers create the application by modeling it to our specifications in order to reach satisfaction with our users.
Develop Application
System
The application provides contextual information for the contributors. It also is the work environment we will be working with as developers.
Google API
PayPal API
Bill Calculation Algorithm
Support the App
Contributors
Be a driver and get paid for rides.
Be a passenger and pay for rides.

Usage Model

The Usage Model is a visualization of all working parts of our application. We wanted to create a diagram that shows our use cases working seamlessly together along with the APIs we wish to use. The diagram starts with the user creating a new trip. This starts the process of selecting the contacts, creating the destination, and choosing the payment. The system calculates the bill, sends the bill, and saves important information to a local file. The application uses two main APIs: Google Maps and PayPal. Google Maps will calculate the miles and use it to calculate the bill in the end. At first, we will only be using PayPal as the main payment service and will add more payment services later. Below, we have provided a Usage Model for the overall working system.



Use Cases

Create New Trip

Description	Creates a new trip
Used by	Users
Preconditions	User starts app
Success End Condition	Creates a new blank template to work with
Failed End Condition	Does not create a blank template
Actors	1. Users that want to rideshare
Trigger	User selects “New Trip” button
DESCRIPTION	
	1. System presents front page 2. User selects “New Trip” 3. System creates a new blank template and saves to database 4. System send user to the new template’s main page
EXTENSIONS	
	None
EXCEPTIONS	
	None
Related Information	Priority: Low Priority Performance: Instant Frequency: Very Often Channel to Primary Actor: Interactive
Open Issues	1. How do we create and save the new template?
Schedule	December 2019

Select Contact(s)

Description	Select contact(s) from the contact list
Used by	Users
Preconditions	User selects “New Trip”
Success End Condition	Contacts are added to the list
Failed End Condition	Does not add contacts
Actors	1. Users that want to rideshare 2. Phone contact information
Trigger	User selects “Add Riders” button
DESCRIPTION	
	1. User selects “Add Contact” 2. System presents popup message asking for access to phone’s contact information 3. System transition user to phone’s contact list 4. User picks contact(s) 5. System saves contact information to list 6. User press “Done” to finish 7. Return to main page
EXTENSIONS	
	1: User can manually add contacts instead of getting it from the contact list 4: User can delete contact(s) from list • Select a contact, then select “Delete” 5: User chooses “Cancel” • Deletes all contacts and sends user back to the main page
EXCEPTIONS	
	1: At the start of the app, the same popup message is shown • If the user already accepts the popup, then don’t show.
Related Information	Priority: Medium Priority Performance: Depends on user Frequency: Often Channel to Primary Actor: Interactive

Open Issues	1. How do we get the contact information and save it to the database?
Schedule	December 2019

Create Destination

Description	Creates destination
Used by	Users
Preconditions	User selects "Destination" button
Success End Condition	Send user to Google Maps
Failed End Condition	Does not send user to Google Maps
Actors	1. Users that want to set a destination 2. Google Maps API
Trigger	User create "stops" in Google Maps
DESCRIPTION	
	1. User selects "Destination" button 2. System sends user to Google Maps 3. User create destination using Google Maps 4. Google Maps calculate miles 5. User selects "Finish" 6. System retrieve miles 7. System sends user back to main page
EXTENSIONS	
	2: User can input more than one destination called "stops"
EXCEPTIONS	
	5: User can select "Cancel" instead to exit • System will send user back to main page
Related Information	Priority: Highest Priority Performance: Depends on user Frequency: Very Often Channel to Primary Actor: Interactive
Open Issues	1. How do we get Google Maps API to work? 2. How can we retrieve the data from Google Maps?
Schedule	December 2019

Calculate Bill

Description	Calculates bill using average gas cost and mileage
Used by	System
Preconditions	User finishes creating the destination
Success End Condition	Calculates an estimate of the cost of the trip
Failed End Condition	Does not calculate correctly
Actors	1. System
Trigger	Automatically after user finishes creating the destination
DESCRIPTION	
	1. System sends user to main page after finishing creating the destination 2. System grabs the average gas price 3. System retrieves the mileage from the destination 4. System uses both to calculate the bill 5. System presents the bill to the user
EXTENSIONS	
	None
EXCEPTIONS	
	None
Related Information	Priority: Medium Priority Performance: Instant Frequency: Often Channel to Primary Actor: Interactive
Open Issues	1. How do we get the average gas price?
Schedule	December 2019

Connect to Paypal

Description	Connect user to Paypal
Used by	Users
Preconditions	User has a Paypal account
Success End Condition	User successfully connects to their Paypal
Failed End Condition	Does not connect successfully
Actors	1. Users who have a Paypal account 2. Paypal API
Trigger	User selects "Connect to Paypal" button
DESCRIPTION	
	1. User selects "Connect to Paypal" 2. System opens Paypal 3. Using Paypal API, users can connect to it.
EXTENSIONS	
	None
EXCEPTIONS	
	3: Users who do not have an account will not be able to connect
Related Information	Priority: High Priority Performance: Quick Frequency: Not Often Channel to Primary Actor: Interactive
Open Issues	1. How do we use the Paypal API?
Schedule	December 2019

Choose Payment Options

Description	User can choose payment options
Used by	Users
Preconditions	The bill is already calculated
Success End Condition	Users have the flexibility of payment options
Failed End Condition	Users do not have any payment options
Actors	1. Users who want to choose other payment options
Trigger	User selects “Payment Option” button
DESCRIPTION	
	1. System calculates the bill 2. User selects “Payment Option” 3. User have the option add/drop/alter the bill
EXTENSIONS	
	None
EXCEPTIONS	
	None
Related Information	Priority: Medium Priority Performance: Depends on user Frequency: Often Channel to Primary Actor: Interactive
Open Issues	
Schedule	December 2019

Send Bill

Description	Send bill to both parties
Used by	Users
Preconditions	When destination is met
Success End Condition	Send bill to everyone involved
Failed End Condition	Does not send bill
Actors	1. User who wish to send bill
Trigger	User selects “Send Bill” button
DESCRIPTION	
	1. User selects “Send Bill” 2. System opens the list containing the contacts 3. User selects who to send the bill to 4. System uses phone number/email to send the bill 5. System informs user, the bill is sent 6. System sends user back to the main page
EXTENSIONS	
	3: User can manually input numbers and emails
EXCEPTIONS	
	None
Related Information	Priority: High Priority Performance: Depends on user, quite fast Frequency: Often Channel to Primary Actor: Interactive
Open Issues	
Schedule	December 2019

Requirements

Functional Requirements

- Google Map API
  - Using Google Maps we will route the trip from A to B.
  - With that route, we can collect the total miles the trip will take.
- PayPal API
  - Access the API and allow the user to enter their Paypal login & password.
  - The application stands-by and wits for the transaction to finish.
- Fuel cost
  - We come up with a price per gallon for fuel.
  - Take an average for Los Angeles county from the internet
    - Web Scraper
  - Developers enter manually on a monthly basis (not the best idea).
  - Allow user to enter fuel cost per gallon that they see advertised.
- Cost calculation logic
  - We calculate a price based on number of occupants, miles driven, miles per given, and the fuel cost.
- Notifications
  - Ideally we would want a driver waiting on payment to notify someone through the application, but that may not be possible as we are not currently requiring accounts be made with us. Instead we will want to access the users contacts list, and notify them through text message instead.
- Quantify Carbon Emission
  - If the user carpooled with 3 other people, show them the difference on the environment a total of 4 vehicles would have had versus only one.

Quality of Life Requirements

- Multiple pay service API
    - While arguably the most widely used, PayPal has competitors and so in an effort to make this application readily available to everyone, we will include other pay services.
- Stat tracking
    - For fun and also to better quantify their lower impact on traffic and parking congestion, keep track of different stats:
        - Number of trips
        - Average number of people in the carpool
        - Miles carpooled
        - Average paid per trip
            - And many more
- Application must be private
    - User’s location information will be secure

Constraints

- Storing data for contact’s bill should be stored on a server instead of locally
- APIs have a limited use rate

Development Process

- Document all steps
- Implement framework
- Set up API tokens
- Set up user interface
- Implement and test basic functionality
- Continue to add quality of life improvements