|  |  |
| --- | --- |
|  | “Flock”: A rideshare application |
|  |  |
| 10/16/2014 | Requirements Document |
|  | Flock shall provide users with the intention of participating in shared rides, through either request or offer with a personal vehicle, the interactive technology and rideshare platform supporting a mobile forum facilitating connection. |

Table of Contents

[Introduction 3](#_Toc401267202)

[Glossary 3](#_Toc401267203)

[1 General 4](#_Toc401267204)

[1.1 Project Description 4](#_Toc401267205)

[1.2 Points of Contact 5](#_Toc401267206)

[1.3 References 5](#_Toc401267207)

[2 Functional Requirements 6](#_Toc401267208)

[2.1 Application Interface/Interactions 6](#_Toc401267209)

[2.2 Client/Server Interaction 7](#_Toc401267210)

[2.3 Use Cases 8](#_Toc401267211)

[2.3.1 System Common Utilities Use Case 8](#_Toc401267212)

[2.3.2 System Server Use Case 9](#_Toc401267213)

[2.3.3 Post Request Activity Diagram 10](#_Toc401267214)

[3 Non-Functional requirements 11](#_Toc401267215)

[3.1 Security 11](#_Toc401267216)

[3.2 Data Currency 11](#_Toc401267217)

[3.3 Reliability 11](#_Toc401267218)

[3.4 Recoverability 12](#_Toc401267219)

[3.5 System Availability 12](#_Toc401267220)

[3.6 Fault Tolerance 12](#_Toc401267221)

[3.7 Performance 12](#_Toc401267222)

[3.8 Capacity 12](#_Toc401267223)

[3.9 Data Retention 13](#_Toc401267224)

[3.10 Environment 13](#_Toc401267225)

[4 Introduction to the User Manual 13](#_Toc401267226)

[Index 14](#_Toc401267227)

“Flock”: A rideshare application

Requirements Document

#### Purpose of Document

This document is a formal statement of the Flock application’s requirements: descriptions of what it does, the services that it provides, and the constraints on its operation. This document should be used for the following purposes:

1. Designing and developing flock rideshare application system.
2. Evaluating flock in all phases of its development.
3. Measuring the success of the project result.

# Introduction

Flock shall enable likeminded individuals wishing to participate in a culture of community rides in which personal vehicles are used to fulfill transportation requirements, by providing the interactive technology and ridesharing platform to connect.

## Glossary

**Flock**: A rideshare application

**SOAP:** Simple Object Access Protocol

**JSON:** JavaScript Object Notation

**SQL Injection:** a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution

**PPA:** The Philadelphia Parking Authority

# 1 General

## 1.1 Project Description

Flock shall enable likeminded individuals wishing to participate in a culture of community rides in which personal vehicles are used to fulfill transportation requirements, by providing the interactive technology and ridesharing platform to connect.

#### Background

Ridesharing in Philadelphia has not flourished as it has in other cities like Chicago, San Francisco or Washington DC. Several factors have contributed to the demise of all previous attempts but the most notable are legislation, the mighty Philadelphia Parking Authority (PPA) ruling over the taxi and limousine industry, and city culture. Including a lack of true adhesion and community among the residents of the differed and many parts of the vast city. In Washington, DC, for example, a popular rideshare application known as “Slug” is widely utilized by suburbanites on a daily basis to achieve their commute into the government center. The reason “Slug” was adopted so rapidly by residents of the greater metropolitan area is because of the single shared industry of employment amongst its residents; almost everybody works for the federal government. Believing that rideshare can be successful in Philadelphia as well, by finding the right cultural glue to bind together everyone from any part of Philadelphia to the Main Line, and by bypassing the rules and regulations of the PPA, we introduce “Flock” a rideshare application for the city of brotherly love, designed specifically for its residents and its laws. We shall call this application “Flock” in honor and symbolic fashion of geese, who participate in great migrations up and down the east coast and through our city twice a year.

#### Purpose

To introduce a mobile application specifically geared toward developing a community of carpoolers in the City of Philadelphia by providing the incentive of community and fellowship while also providing a sense of security to the participants.

#### Assumptions

All participants will provide certain personal information before using the application. All personal information will be kept secure and anonymous except in the event of the participant being involved in a criminal act while using the application. No one using the application will have financial gains except for reimbursement to drivers for all fuel-related expenses.

#### Constraints

All attempts will be made to avoid falling under jurisdiction of the PPA because to do so would mean quick conversion to a business model that is not a rideshare but instead a taxi service or probably more likely, a swift and sudden shutdown. Therefore we must be certain that “Flock” cannot be used to operate taxi services, or more specifically, rides for money beyond reimbursement for fuel.

## Points of Contact

* JiaNing Li, jl592267@sju.edu
* Nathan Otto, no564972@sju.edu
* John Candelori, jc417242@sju.edu
* Wensheng Yan, wy590204@sju.edu
* Kate Portalatin, kp593753@sju.edu

## References

1. Client use SOAP to communicate with Server

KSoap2 <http://code.google.com/p/ksoap2-android/>

1. Find nearest driver or passenger (like store locator)

<https://developers.google.com/maps/articles/phpsqlsearch_v3>

1. Using Google Maps in Android

<https://developers.google.com/maps/documentation/android/start#getting_the_google_maps_android_api_v2>

1. Android Location API - Tutorial

<http://www.vogella.com/tutorials/AndroidLocationAPI/article.html#locationapi>

1. google gson

<https://code.google.com/p/google-gson/>

1. android iconography

<http://developer.android.com/design/style/iconography.html>

1. Ridesharing in Philadelphia

<http://articles.philly.com/2014-07-24/news/51956654_1_uber-and-lyft-transportation-network-companies-uberx>

# 2 Functional Requirements

## 2.1 Application Interface/Interactions

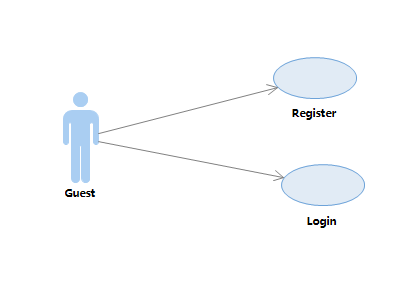
* User is to be prompted with a single login page with a central picture of the Flock logo. Under the logo there will be two text boxes where the user may type in their username and password for their account. Underneath of the text boxes will be a hyperlink asking the user if they are “New to Flock”. Below the “New to Flock?” hyperlink, there will be another hyperlink asking the user if they “Have forgotten their account information”. If the user clicks either hyperlink a small window will appear prompting the user to either create a new account or to answer security questions the user chose when their account was initially created. If questions are answered correctly, the users account information will be sent to their personal email.
* User shall be able to gain access to the system only after properly authenticating with a valid username and a password and providing a current photo.
* User shall not be able to offer or accept rides until certain information is verified. For all users this includes verifying a cell phone number with receipt and a reply of a text message. Users will be required to verify an email address with receipt and reply, and for drivers it will also include verifying their driver’s license id number with the Department of Motor Vehicles.
* User password will be encrypted using MD5 algorithm before it sends to server.
* Upon entering and verifying the information, the data will be sent to the server and authenticated.
* A user’s unique identifier is their chosen userid.
* Users who are authenticated will be brought to the main function screen where they will be able to search for a new Flock ride using the map.
* Upon initial login, users who are both Riders and Drivers will select either a Rider or Driver in the top right portion of the screen depending on the role they are fulfilling at that specific time.
* The Application will allow users to view their profile as well as logout by clicking the menu button or swiping the menu out from the left side of the application.
* The menu will provide the user the ability to view their profile picture, edit their profile, and perform other desired functions within the application.
* Current location shall be shown on the map, given that location services are turned on the device. This is to be determined using a Google Maps API that is available.
* Riders shall enter desired location in the search box on the screen. The box will allow for users to enter in location names and the system will use Google Maps to search for a valid destination on the map.
* Drivers shall be able to set the destination for their current Flock trip using the same destination box on the screen.
* Riders shall be able to select Drivers and view the number of available seats in the vehicle.
* A count will be kept in the database that will be updated each time a Rider is added to the Driver’s vehicle.
* Drivers with full occupancy shall not be able to accept any more Riders in the system and will not appear on Rider’s map.
* Riders shall be able to select Drivers by clicking on their icon and clicking “REQUEST RIDE”. From here a request is sent to the corresponding Driver to be able to “accept” or “decline”.
* Pairings shall only be made by the system if it is mutual between Rider and Driver.
* There will be a function that calculates the potential reimbursement cost for fuel based on mileage and how many riders are in the vehicle. Each time a participant accepts a ride, he/she must also accept the potential reimbursement fee for which the drive has a right to ask. Each time a rider is added to the vehicle, that fee goes down a little for each rider.

## 2.2 Client/Server Interaction

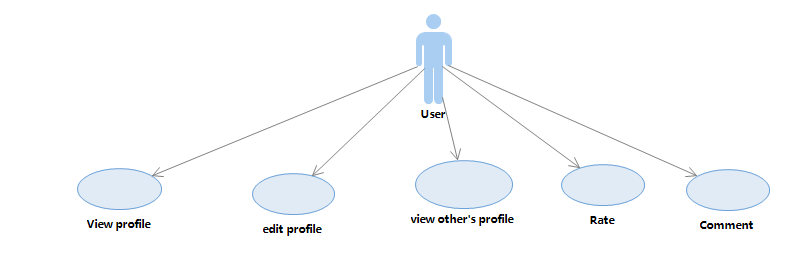
1. Client and Server should communicate based on SOAP protocol.
2. Server should authenticate username and password before user post requirement or search list from the server.
3. All object should be parsed to JSON language before it sends to another side.
4. Server should verify all the user’s submitting data to make sure it is valid.
5. Server should filter specific characters to avoid SQL Injection attack.
6. Sever will generate a unique session id when user who are authenticated.
7. When server receive client’s request, it should authenticate user’s session, permission before response to client. If it is an invalid request, server should response an error message.
8. Server should search user’s request and return the matching results back to client.
9. Server should help driver calculate price base on the mileage.
10. Server should save all user requests but it should avoid duplicate request.

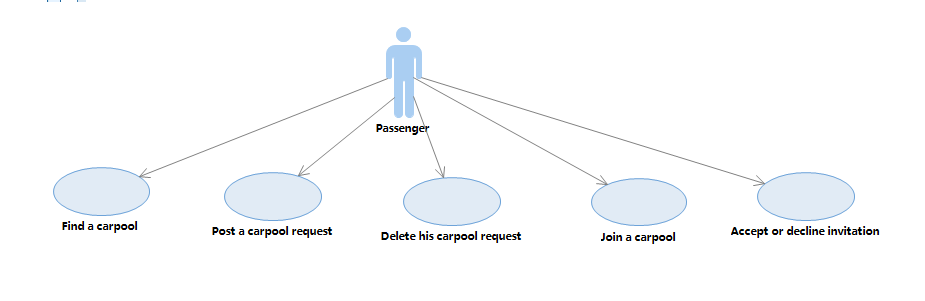
## 2.3 Use Cases

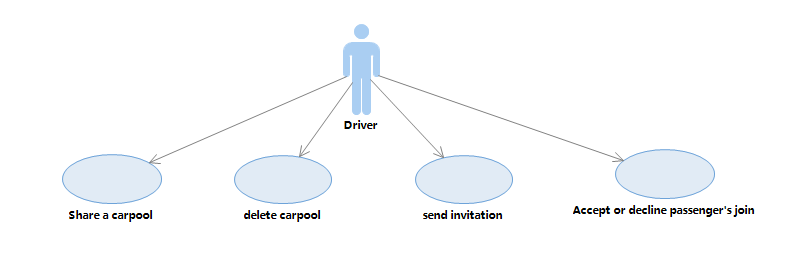
### 2.3.1 System Common Utilities Use Case



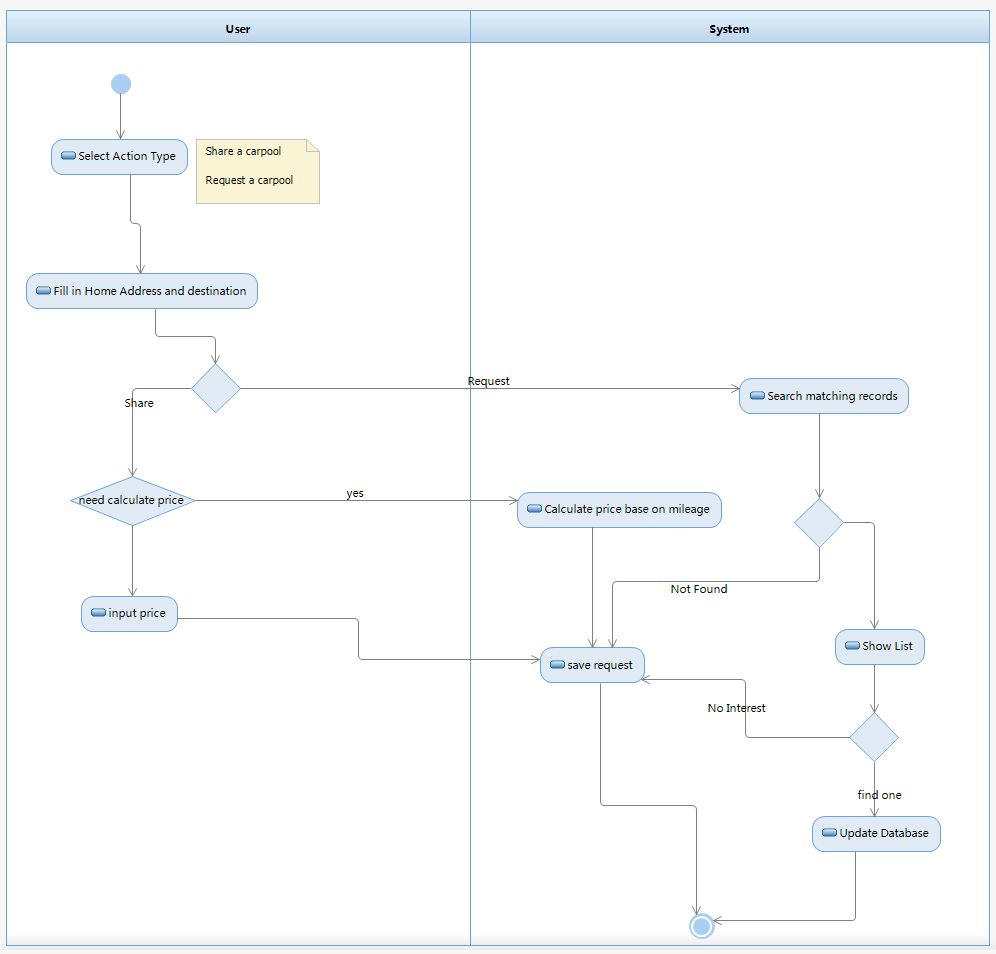
### 2.3.2 System Server Use Case







### 2.3.3 Post Request Activity Diagram



# 3 Non-Functional requirements

## Security

Security is a particularly important element of the software. As we have discussed as a group on more than one occasion, there needs to be a certain amount of trust amongst its user in order to be successful, and there is potential for “Flock” to be used maliciously to commit crime. We need the user to provide some private information in order to track users of the application and to make it difficult for anyone to use the service maliciously and get away with it, as well as to provide reassurance and security to the majority of well-meaning users. It is also important that we keep users data perfectly secure to provide a certain degree of anonymity and protection from people who may wish maliciously hack into our system. So, the security issue is two-tiered and we will remain vigilant on both levels. For example, we will require that drivers provide a valid US driver’s license number as well as the make, model and year of their car and their vehicle tags. Every user of the system must provide “Flock” with their full legal name, a current valid address, and a valid phone number and email address which must be verified before use of the application. Every user must provide a photo of themselves as their profile identifier but they are not required to reveal any personal information to each other beyond a photo. Only those developers who have a need to know should have permission to view private information stored on the database as well as the MD5 algorithm that protects the password.

## Data Currency

We want to develop software that lets users find other users nearby with similar destination goals, and the currency should be rapid to the users. For example, when a user begins a search, he/she should receive information that is current up to 1 minute to avoid having multiple instances of a rider requesting a driver who is no longer available. After each new search and each refresh the speed of the feedback should be delivered in several minutes. Sorting should be available to help the user find the request information quickly.

## Reliability

If the software goes wrong and collapses, it should not cause much trouble. If the user enters an address and the application cannot find it, this would be annoying to the users, and so all maps should be reliable and intuitive, and able to find locations even when there is incorrect spelling or incomplete entries of locations.

As for the database, our program needs to have quick and frequent access to the database at all times, from the searching to the accepting of an offer. If the database is down, the user should not have long to wait for access.

The mean time between failures should be no more than 24 hours. If the user finds the software malfunctioning more than once in a day, he will choose to not use the application. Usually, the common way to repair software is to restart, so we shall shorten the restart time as much as possible, and in under one minute.

## Recoverability

At every use of the application the database will be tapped, and so any corruption of the database would cause a lot of troubles. Database recovery should be rapid and at the most one hour.

In the event the application is unavailable, the user may choose not to use it. If any one part of the software is malfunctioning or producing incorrect data, the entire application will be useless. If the application is unavailable, it should be repaired within 1 day. Each day the regular user wakes up to a malfunctioning application, the more annoyed he/she will become and the more likely he/she is to abandon its use altogether.

## System Availability

It is important that the system’s calendar and clock information remain consistent in order to arrange accurate and prompt meetings amongst its users. The software will download the current time each time it connects to the database, so the system administrator must ensure that the database time is always correct.

## Fault Tolerance

Searching, reading and accepting are all vital parts of the application and if one of them fails, the whole program will fail. All functions should be tested and analyzed often in order to avoid fault in the application. If the system encounters a fault, a warning and notice to the system administrator shall be posted immediately.

## Performance

When searching another user’s offer or request, the response time should be within one minute. When he/she reads the information and accepts it, we must avoid the possibility that the driver has already departed for the destination. The update will be much easier for searching queries, so it also shall not take more than one minute.

The volume of data will be large. Yelp has a monthly average of 78 million unique visitors. This program will not work like yelp but even if we do 1% of Yelp’s volume, we would have 780 thousand visitors and require nearly 1.5G of database, assuming the average user sends three requests per month. This figure is based on calculating the assumptions that each request requires at least 2 addresses, in addition to the user’s two preferred addresses, or about1600bytes. There will be some other information we need to record, so one person could take as much as 2000 bytes. If we multiply 2000 by 780 thousand, the result is 1560 GB.

We also must consider that in one request we need to show the position using the map function and the data flow may be 10k-100k, making the total flow per month 7G-70G.

## Capacity

Refer to the calculation listed in section 3.7. The request part of a user function can be deleted every month or every year, but user’s personal information shall be kept indefinitely. Considering that the number of users could grow exponentially, the database should be two to three times larger than our calculated need, so we will require a 3-5G database.

## Data Retention

Discussed in section 3.7. In this case, monthly retention is enough. Assuming the needs of the average user to remember rides up to one month in the past, we will provide user a record of his/her personal use for the thirty days prior to the current date.

## Environment

|  |  |
| --- | --- |
| Operating System | Server: Linux, Client: Android 4.2 or higher |
| Database | MySQL 5 or higher |
| WEB Server | Apache 2, PHP 5.3 or higher |

# 4 Introduction to the User Manual

1. Access to the android application shall be available for anyone running 4.2 Jellybean or higher.
2. Each user in the system shall be uniquely identified by their email address used upon registration regardless of if they are a rider or a driver.
3. The same account will be used for an individual who is both a rider and a driver.
4. Upon authenticating for the first time, the user will be asked to create a profile which will consist of the following information:

For Riders:

Display Name

Profile Picture

Current Address

Phone Number (must be verified)

Email Address (must be verified)

For Drivers:

Display Name

Profile Picture

Car make

Car model

Car year

Number of Seats

Driver’s License ID number

License plate tag

1. Once on the main screen, the Rider will be able to input into the designated field their desired destination.
2. Available Drivers who are traveling within a 1 mile radius of the desired location, will be displayed on the map (Google Maps API will be used).
3. A Rider shall be able to tap on driver’s icon to view the individual’s profile and driver ratings.
4. A Rider must accept a request for fuel reimbursement before accepting a ride.
5. A Driver is able to post a monetary request with the applications fuel calculator function for a ride being offered to a certain location that will not exceed the cost of fuel.
6. A Driver shall be able to set the location for their current Flock trip, in order to pair with Rider’s who are
7. Upon selecting a desired driver, a meeting location is decided between the Driver and the Rider.
8. After the ride is completed, the Rider and Driver will both be prompted to rate the Flock trip on a scale of 1-5 stars, and leave a comment about the other user up to 500 characters.

# Index

Authenticate, 6, 7

Capacity, 11

Data Currency, 10

Data Retention, 12

Environment, 12

Fault Tolerance, 11, 12

JSON, 3, 7

Performance, 10

Recoverability, 11

Reliability, 10

Requirements, cover, 3

Functional, 6

General, 4

Non-Functional, 10

Security, 4, 10

SOAP, 3, 5

System Availability, 11

Use Case, 7, 8

User Manual, 12