

6-5-2015

# Wakabi: on-demand ride service for rural Uganda

Michael Brew  
*Santa Clara University*

Bryant Larsen  
*Santa Clara University*

Follow this and additional works at: [http://scholarcommons.scu.edu/cseng\\_senior](http://scholarcommons.scu.edu/cseng_senior)



Part of the [Computer Engineering Commons](#)

---

## Recommended Citation

Brew, Michael and Larsen, Bryant, "Wakabi: on-demand ride service for rural Uganda" (2015). *Computer Science and Engineering Senior Theses*. Paper 37.

This Thesis is brought to you for free and open access by the Student Scholarship at Scholar Commons. It has been accepted for inclusion in Computer Science and Engineering Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact [rscroggin@scu.edu](mailto:rscroggin@scu.edu).

**Santa Clara University**  
*DEPARTMENT of COMPUTER ENGINEERING*

Date: June 5, 2015

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY  
SUPERVISION BY

**Michael Brew and Bryant Larsen**

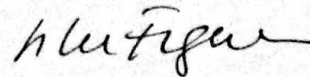
ENTITLED

**Wakabi: On-Demand Ride Service For Rural Uganda**

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING**



\_\_\_\_\_  
THESIS ADVISOR

  
\_\_\_\_\_  
DEPARTMENT CHAIR

# **WAKABI: ON-DEMAND RIDE SERVICE FOR RURAL UGANDA**

by

Michael Brew and Bryant Larsen

## **SENIOR DESIGN PROJECT REPORT**

Submitted in partial fulfillment of the requirements  
for the degree of  
Bachelor of Science in Computer Science and Engineering  
School of Engineering  
Santa Clara University

Santa Clara, California

June 5, 2015

## Abstract

In Uganda, the majority of the population lives in rural villages that rely on last-mile distribution for goods such as vaccines, fresh water, trade goods, and other forms of humanitarian relief. Last-mile distribution refers to the last mile (or few miles) that goods must be transported in order to reach their final destination from a main delivery hub. Coordination is one of the primary issues that exist when trying to solve the last-mile problem. In this paper we present our solution to this problem: an SMS-based, on-demand ride-sharing service designed to empower the people of rural Uganda by helping organize the transport of both people and goods.

Our application functions similarly to the popular ride-sharing app Uber or Lyft but does not require a smart-phone to use. Users text a predefined number to request a ride, get paired with a nearby boda-boda driver (these motorcycle drivers currently offer ride-sharing services to rural Ugandans by word-of-mouth), and are transported to their destination. The service also allows users to specify trailer requirements in case they need to transport goods as well. By building Wakabi around the existing boda-boda system we are not only helping to coordinate last-mile distribution efforts, but are also improving the efficiency of the existing boda-boda drivers that provide transportation to rural Ugandans. Following the 2014-2015 academic year, Fulbright Scholar and business partner Ty Van Herweg will be responsible for both testing and deploying Wakabi in Uganda. We hope that our application will help boda-boda drivers better serve their riders, and provide businesses with an ideal and cost-effective last-mile distribution solution.

## Acknowledgments

We would like to acknowledge the following individuals and institutions for helping us reach our goal of completing a fully functional, on-demand ride service accessible through SMS. They have provided valuable advice, feedback, ideas, and resources that were necessary to our mission.

### **Professor Silvia Figueira**

Associate Professor of Computer Engineering and Director of the Frugal Innovation Lab at Santa Clara University;  
Wakabi Senior Design Advisor

### **Tyler Van Herweg**

Fulbright Scholar, SCU class of 2015 (Economics and Theatre double-degree, English minor);  
Original inventor of Wakabi idea

### **Heidi Williams**

Director of Communications, SCU School of Engineering;  
Provided publicity through interview and photoshoot with the Wakabi team

### **Friends and Family**

For helping us test our service as well as providing useful feedback for improvements

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem Statement	1
1.2	Background & Motivation	2
1.3	Objectives	2
1.4	Literature Review	3
<b>2</b>	<b>System Design</b>	<b>4</b>
2.1	Requirements	4
2.2	Use Cases	5
2.3	Conceptual Model	9
2.4	Architecture	13
2.5	Technologies Used	14
2.6	Design Rationale	15
<b>3</b>	<b>Testing</b>	<b>17</b>
3.1	Test Plan	17
3.2	Test Results	19
3.3	Future Testing	19
<b>4</b>	<b>Societal Issues</b>	<b>20</b>
4.1	Ethical	20
4.2	Economic	20
4.3	Health and Safety	21

4.4 Usability	.....	21
<b>5 Conclusions</b>		<b>22</b>
5.1 Maintenance Guide	.....	22
5.2 Suggested Changes	.....	24
5.3 Lessons Learned	.....	25
<b>6 References</b>		<b>26</b>
<b>7 Appendices</b>		<b>27</b>
7.1 Activity Diagram	.....	27
7.2 Entity-Relationship Diagram	.....	29
7.3 Component State Diagram	.....	30

## List of Figures

2.1	Use Case Diagram	5
2.2	Requesting Rides Screenshot	9
2.3	Available Drivers Screenshot	10
2.4	No Available Drivers Screenshot	10
2.5	Positive Feedback Screenshot	10
2.6	Negative Feedback Screenshot	10
2.7	Accepting Request Screenshot	11
2.8	Rejecting Request Screenshot	11
2.9	End Ride Screenshot	11
2.10	Start Shift Screenshot	12
2.11	End Shift Screenshot	12
2.12	Architecture	13
7.1	Activity Diagram	27
7.2	Entity-Relationship Diagram	29
7.3	Component-State Diagram	30



## List of Tables

3.1	Requesting Rides	.....	17
3.2	Reviewing Drivers	.....	18
3.3	Changing Driver Availability	.....	18
3.4	Start/End Rides	.....	18
3.5	Responding to Ride Requests	.....	19

# Chapter 1. Introduction

## 1.1 Problem Statement

Entrepreneurs recognize that time is money and that any time or effort wasted in last-mile distribution takes away from their businesses. In Uganda, 85% of the population lives in rural areas [1] and many people do not have the means to transport either themselves or their goods to where they need to go. Farmers need to move their produce, business people may need to work in big cities, and even non-governmental organizations or charities need to ship their supplies to those in need. These objectives should not be impeded by lack of transportation or inaccessibility to individuals who live in remote areas.

In Uganda, there are currently fleets of hireable motorcyclists that will ferry people and goods between remote villages and big cities, but connecting drivers and riders is a difficult process that decreases the potential efficiency of the entire system. The service relies on word of mouth and does not provide a unified interface for riders to request drivers, leaving many villagers helpless. On the other hand, drivers spend much of their time waiting around for potential customers because they generally only have access to a limited pool of clientele.

Our mobile solution allows people needing rides to connect with drivers who are in need of clients via SMS. Users can text a predefined number with their locations and whether they need to transport goods or just themselves. A motorcyclist registered with our service then receives a text if he or she is in a nearby location and responds by either accepting or rejecting the request. If accepted, the rider's phone number is sent to the driver so that a meeting location can be established. These drivers pay a monthly fee to be registered with our service while collecting their fare directly from the riders.

A system that connects people who have logistical needs with available motorcyclists through SMS will open the door for many of those that were previously bound by their lack of transportation. Drivers will spend less time idly waiting, instead receiving texts whenever someone needs a ride, thereby increasing their profit. With the possibility of attaching trailers to the motorcycles, large quantities of goods can be easily moved on-demand. We are not giving the people of Uganda new modes of transportation, but are connecting them to cooperatively help achieve each other's goals.

## 1.2 Background & Motivation

We first found out about this project from our advisor, Professor Figueira. She told us that a senior business major here at SCU had previously traveled to Uganda and had a vision to return with an SMS-based on-demand ride service. That business major and partner, Tyler Van Herweg, had spent the summer in Uganda in 2013 thanks to SCU's Global Social Benefit Fellowship. He worked with a local company called BanaPads that produces women's sanitary pads from banana fiber. He gathered a great insight into Ugandan life, especially in Kampala (the capital) and the surrounding rural area. He noted the transportation situation and the high demand to move people and goods, whether it be farmers needing to transport their crops to Kampala, or people simply needing to reach other villages.

Here in the US, we have used similar services offered through smartphone apps, such as Uber and Lyft. However, we chose not to develop a smartphone app because the people we are trying to serve in rural Uganda do not generally have access to apps. However, Tyler noticed that many of the people he worked with and met did have a feature phone that had SMS and call abilities. Therefore, by creating a service accessible completely through SMS, we can reach as many people in our target audience as possible.

Rather than providing a completely new service (complete with resources and personnel), we want to actually help both drivers and potential riders currently in Uganda. Since there are already hireable motorcyclists called "boda boda" drivers, Wakabi can provide a simple "funnel" for anyone needing a ride to any available boda drivers. There is a similar service currently deployed within Kampala called "Safe Boda" that allows you to request a boda driver from a smartphone app. But as previously mentioned, we want to have an SMS based service because the people of rural Uganda do not have as many smartphones as the people of Kampala.

## 1.3 Objectives

The main objective of our project was to create a robust, easy-to-use application that will make it easier for individuals to obtain transportation within rural Uganda. The application's purpose is to provide a service that will improve the system of ride-sharing that already exists in Uganda. Our project will achieve this by seamlessly connecting drivers and riders using a standard communication interface. We are also trying to achieve maximum accessibility with the project's distribution, which is why we chose to

implement the project using SMS. Additionally, simple keywords, presented in the native language of Luganda, allow us to reach users with limited literacy.

Our main objectives can be summarized in the following points:

- Provide an accessible and affordable option for local entrepreneurs to aid in product delivery
- Provide an innovative solution to larger businesses to aid in last-mile distribution
- Increase customer traffic for Boda drivers and thereby cut down their idle waiting time and increase their economic efficiency
- Provide a transportation solution for everyday consumers in rural Uganda that need to travel between village to village or to larger cities

## 1.4 Literature Review

Prior to starting development, we spent time to research how an SMS-based mobile service could impact rural Ugandans as well as how receptive our target audience might be. Kas Kalba [2] provides an insightful article “Africa’s Mobile Money Story” in *Intermedia* about how the use of “Mobile Money” has expanded within Africa in the past 6 years. His analysis shows that services offered through mobile technology have been well received by the African population. If users are comfortable trusting SMS to handle their banking transactions, hopefully they will similarly trust SMS to handle their transportation as well.

This data is supported by an article in *BMC Medical Informatics & Decision Making* [3] that reports the findings of a survey asking Ugandans if they would support an SMS based service that relays personal medical information. The majority of participants said they would use such a service, while 90% said they were unconcerned about unintended disclosure. These results gave us more trust in building an SMS service that will be deployed specifically to the Ugandan market.

Lastly, we spent some time researching the Boda-Boda driver system existent in Uganda as Wakabi will be servicing these drivers in addition to the general public. An article from *Development Southern Africa* [4] includes survey results concerning Boda demographics and their feelings on their situation. The majority of drivers are young males between 16-30 years of age, and many reported that they feel it is impossible to own the actual motorcycles they use. Hopefully with the use of Wakabi, we can provide more customers to these drivers, thereby increasing their income and overall livelihood.

## Chapter 2. System Design

### 2.1 Requirements

Our solution's capabilities can be broken down to the following requirements, which are organized in three categories: functional requirements, non-functional requirements, and design constraints. The functional requirements define what our system must do, while the non-functional requirements define the manner in which our system must achieve the functional requirements. The design constraints are the restrictions on the implementation of our system.

#### Functional Requirements:

- The system will allow riders to choose between a normal motorcycle or a motorcycle with a trailer attachment when requesting rides.
- The system will prompt riders to indicate their location when requesting rides.
- The system will send a rider's request to the closest driver. If declined, the request will be sent to the next closest driver, and so on.
- The system will alert a rider if no drivers are available and will automatically text the rider if a driver becomes available within the next 30 minutes.
- The system will prompt riders for feedback about their drivers after their rides.
- The system will keep a database of registered drivers who will pay monthly to stay listed in our service.
- The system will allow drivers to indicate that they are available to give rides whenever they choose.
- The system will allow drivers to indicate that they are no longer available to give rides if they are not currently giving a ride.
- The system will use simple keywords to allow users to perform actions (such as drivers accepting requests, riders requesting rides, drivers marking themselves unavailable, etc.).
- The system will present riders with a list of locational zones to which they just reply with the associated list number when asking them where they are located.

#### Non-functional Requirements:

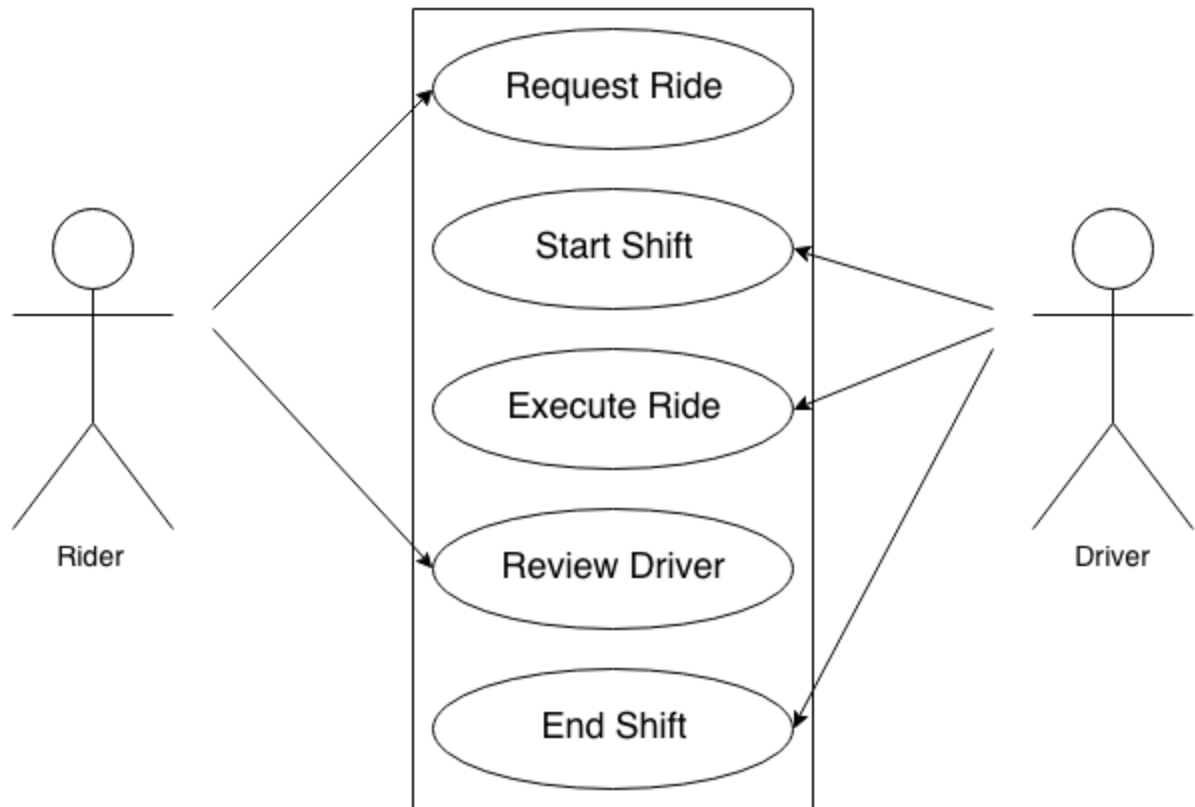
- The system will be easy to use for first-time riders by using minimal instructions.

#### Design Constraints:

- The system must be usable (by both drivers and riders) completely through SMS.

## 2.2 Use Cases

The use cases describe how our system will operate. They include the actors, or the individuals performing the use case, the goals of the use case, conditions that must be true before the use case is performed, the results of the use case, and a description of steps to complete the use case.



1

### ***Request Ride***

**Goal:** To successfully request a ride from the application

**Actor:** Rider

**Preconditions:**

1. Web-server is online and available for client interaction

**Postconditions:**

1. Rider is connected with a driver, or no drivers are available

**Scenario:**

---

<sup>1</sup> Figure 2.1 Use Case diagram depicting use cases and their respective actors.

1. Rider texts the word “RIDE” to our application using SMS
2. System replies to request user location (region)
3. Rider replies with a number corresponding to his/her region location
4. Rider indicates if he/she would like a driver with an attached trailer
5. IF no drivers available, system responds to notify rider
6. ELSE system contacts first driver in same location and sends rider’s phone number

**Exceptions:** User does not have network access to send or receive SMS

### ***Start Shift***

**Goal:** To inform the system that a driver is available to work

**Actor:** Driver

**Preconditions:**

1. Web-server is online and available for client interaction
2. Driver is currently not listed as available

**Postconditions:**

1. Driver has started shift, which adds them to the queue of available drivers

**Scenario:**

1. Driver texts “START SHIFT” to the application using SMS
2. System requests driver location (region)
3. Driver replies with location for work
4. System replies to notify driver of successful request

**Exceptions:** Driver does not have network access to send or receive SMS

### ***Execute Ride***

**Goal:** To assist a driver in providing a ride to a customer

**Actor:** Driver

**Preconditions:**

1. Web-server is online and available for client interaction
2. Driver is currently listed as available to work
3. Rider has already submitted a ride request

**Postconditions:**

1. Rider has successfully completed his/her ride

**Scenario:**

1. System texts first driver in location to inform him/her of a pending ride request
  - 1.1. Driver has the option of rejecting a request, prompting the system to check for the next available driver

2. Once accepted, system texts driver with rider's phone number for pick up coordination
3. Driver texts "END RIDE" to system once destination is reached and end time is recorded
4. System calculates and records total ride time for analytics

**Exceptions:** Driver does not have network access to send or receive SMS

### ***Review Driver***

**Goal:** To allow the rider to provide feedback about his/her driver

**Actor:** Rider

**Preconditions:**

1. Web-server is online and available for client interaction
2. Rider has just completed a ride with one of our listed drivers

**Postconditions:**

1. Rider has sent feedback to the system

**Scenario:**

1. System texts rider asking if he/she was satisfied with their driver
2. IF rider replies YES
  - 2.1. System confirms and records positive feedback
3. ELSE (rider replies NO)
  - 3.1. System replies to request comments about driver
  - 3.2. Rider texts system including a short message that contains driver feedback
  - 3.3. System confirms and records negative feedback

**Exceptions:** Rider does not have network access to send or receive SMS

### ***End Shift***

**Goal:** To inform the system that a driver is not available to work

**Actor:** Driver

**Preconditions:**

1. Web-server is online and available for client interaction
2. Driver is currently listed as available

**Postconditions:**

1. Driver has ended his shift and will no longer receive ride requests

**Scenario:**

1. Driver texts "END SHIFT" to the application using SMS
2. System replies to notify driver of successful request

**Exceptions:** Driver does not have network access to send or receive SMS



To supplement the use cases, we provide an activity diagram (fig. 3) found in *Appendix A*. The diagram is a graphical representation of all the stepwise activities and processes that our system participates in. Many of the use cases are loosely defined within the activity diagram, which attempts to modularize the functions of the project in an easy to read manner.

Additionally, *Appendix C* contains a component state diagram (fig. 5) to further supplement the use cases and activity diagram. The diagram describes the various physical states that our system can exist in, and outlines the processes that change the system state. It is meant to extrapolate the use cases in terms of the physical hardware that will comprise our system.

## 2.3 Conceptual Model

The following sections outline how users can request rides and give feedback, how drivers can accept or reject these requests, and how the driver indicates the start and end of the ride. It also shows how drivers can mark themselves available or unavailable to give rides.

### Notes:

- **All English words presented to users will be translated to Luganda. These user facing strings are stored in a file to make translation easier.**
- **The following figures are shown on an iPhone screen to help visualize the entire conversation in one screenshot, but the service is accessible without a smartphone**

### 1. Requesting Rides (Rider)

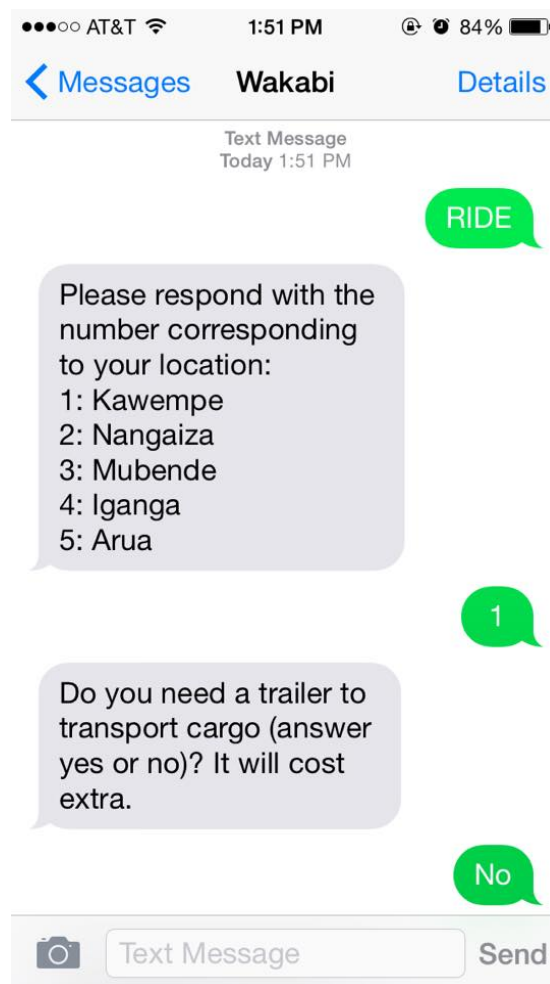


Figure 2.2 Initial transaction to request a ride

1a. *If drivers are available*



Figure 2.3 Response when drivers are available

1b. *If no drivers are available*

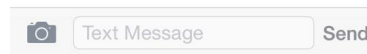


Figure 2.4 Response when no drivers are immediately available

2a. *Giving Positive Feedback*



Figure 2.5 Receiving positive feedback post-ride

2b. *Giving Negative Feedback*

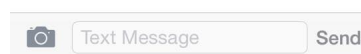
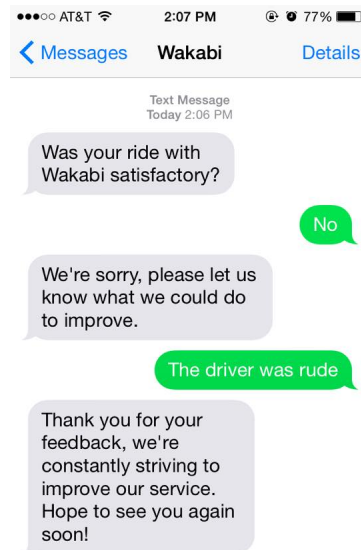


Figure 2.6 Receiving negative feedback post-ride

### 3a. Accepting Request (Driver)

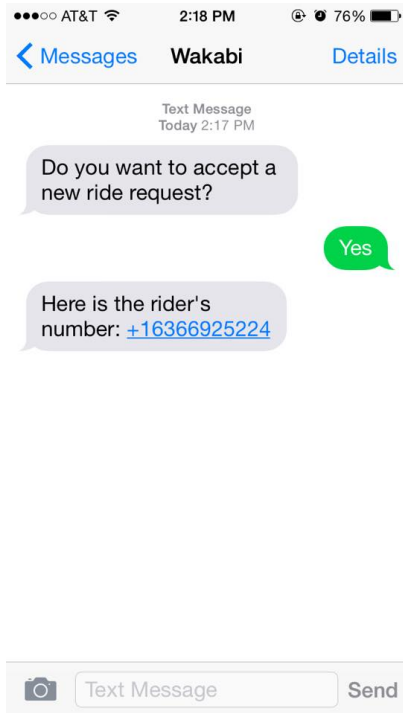


Figure 2.7 Accept a request from the driver side

### 3b. Rejecting Request (Driver)

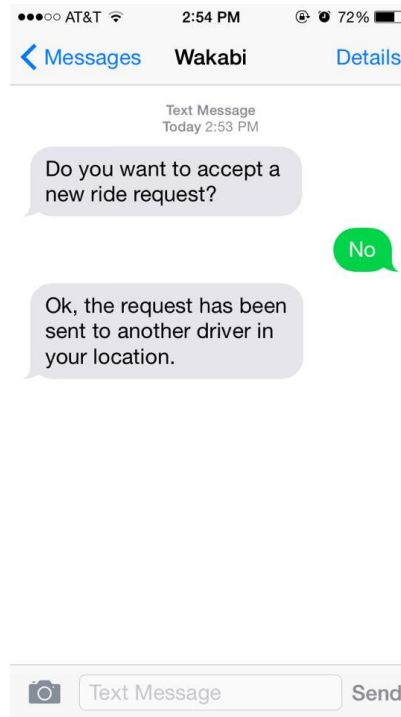


Figure 2.8 Rejecting a request from the driver side

### 4. End Ride (Driver)

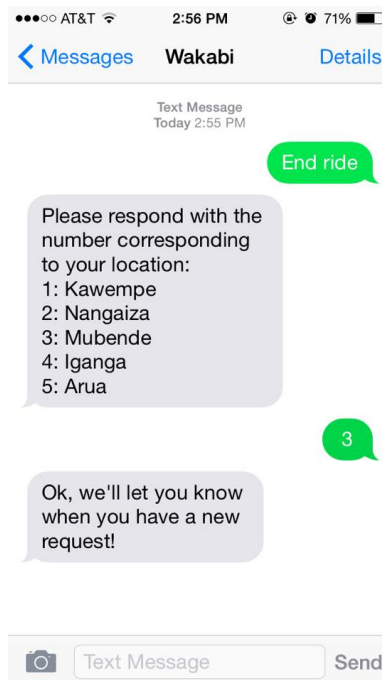


Figure 2.9 Prompt sent to driver after ending ride to track their location

### 5a. Start Shift (Driver)

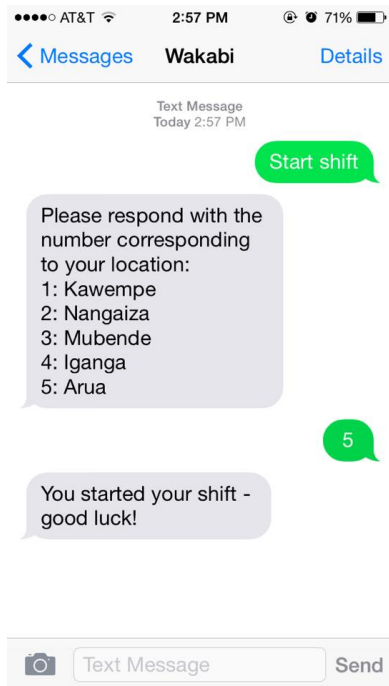


Figure 2.10 Starting shift from the driver side

### 5b. End Shift (Driver)

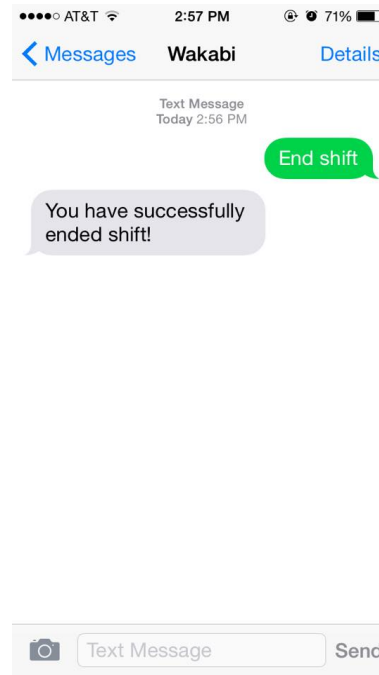
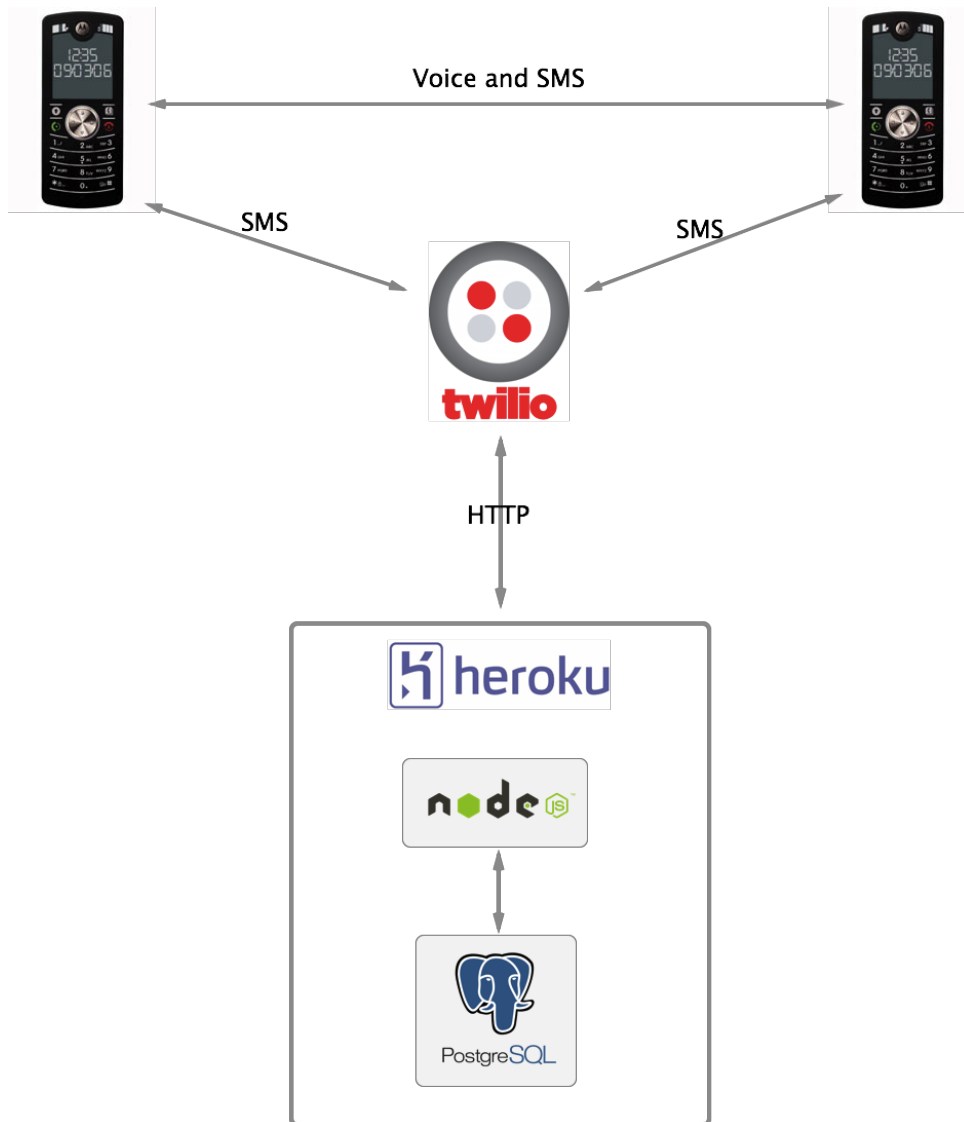


Figure 2.11 Ending shift from the driver side

Additionally, *Appendix B* provides an entity-relationship diagram (fig. 7.2) to aid in the conceptualization of our project. The ER diagram is a graphical representation of entities in our system and their relationships. It also contains the relevant data that we will be storing about our entities, such as driver and rider phone number.

## 2.4 Architecture

Our service will be accessible by SMS to both riders and drivers. As depicted in Fig. 2, the incoming and outgoing texts will be routed through Twilio and processed using the Node.JS framework. Our scripts will be able to communicate with a database to keep track of user (driver and rider) accounts. It will also store information about each ride, like time taken, locations served, driver's name, etc.



2

<sup>2</sup> Figure 2.12 High level architectural design of the system.

## 2.5 Technologies Used

Our mobile service is based almost entirely on SMS, which severely limits its capabilities but ensures any individual with a cellular phone (in Uganda) can access it. The following technologies will be utilized in developing our application. Each technology offers a specific function and is explained in detail below.

### **SMS**

SMS, or short message service, is a service that allows the transmission of text-only messages between mobile phones, fax machines, and/or IP addresses. It is the standard for basic message-sending services between mobile phones. Messages must not be longer than 160 characters in length, and must also not include images or other media. To complete the process, an SMS is sent to a Short Message Service Center (SMSC) and uses a Home Location Register (HLR) to find the roaming customer and deliver the message. SMS offers the simplest communication service for customers to interact with our service via mobile phone.

### **Twilio**

Twilio provides a simple platform to build our SMS application. This service can route incoming calls and SMS messages to a configurable URL over HTTP. Additionally, once the message is handled, it can then send your response back to the original sender. The Twilio service acts as a middle-man between our application server, which is hosting the application scripts and database, and our mobile customers.

### **HTTP**

Twilio uses HTTP to communicate with our externally-hosted web server. HTTP, or Hypertext Transfer Protocol, is an application protocol that is widely used on the Internet to exchange or transfer hypertext. HTTP uses structural text to communicate and relay statuses between clients. For example, when you enter the URL of a website in your browser, you are sending an HTTP GET request for that particular webpage. The server will determine if you are able to access the file before sending a structured HTTP response.

### **Heroku**

Heroku is a cloud-based service for server hosting. We can host our code and database with Heroku, which then provides URL access points for us to utilize. We can then configure Twilio to send incoming texts to our provided URL which will hit our server running on Heroku.

## **Node.JS**

Node.JS is a run-time server environment that we use on our Heroku server. Using the Express web-framework node module, we can accept incoming HTTP requests and send out HTTP response (to and from Twilio). All coding within our Node.JS environment is written in JavaScript.

## **PostgreSQL**

PostgreSQL is our relational database management system for keeping track of rider and driver information. It is offered as an add-on through the Heroku service, allowing for easy integration with our app. It uses standard SQL for queries (also used with MySQL, Oracle, SQLite, etc.).

## **2.6 Design Rationale**

Our entire service relies on the SMS communication between user and server. Therefore, we need to ensure a reliable way to allow easy contact between our system and the drivers and riders. The only exception is that once a driver has accepted a request, the rider's phone number will be sent to him or her so they can call the rider and organize a pickup location. Without GPS technology, determining location and setting meeting spots was not possible via SMS. We also needed to make sure we are handling and responding to our users' messages accurately, or else the consequences may be leaving a potential user stranded, leading drivers astray, etc. We will also have to keep up-to-date records on all of our drivers to make sure that they are receiving requests if they are paying us a monthly fee, while also removing any drivers that are providing a poor or dangerous service under our name. The technologies listed in the "Technologies Used" section detail how we tackled these problems, and the following paragraphs explain our choices.

### **SMS**

We want to reach out to as many people as possible while operating within our means. Allowing riders and drivers to use SMS for our service seemed the most appropriate decision as this mobile solution allows for scalability, automation, and responsiveness. We also chose SMS over a smartphone application because feature phones (or "dumb phones") are much more prominently used than smartphones in our target location, so limiting our service to an app would alienate a large part of the demographic.



## **Twilio**

What allows us to provide our service through SMS is the ability to route our users' texts to our servers and respond back with our own texts. We chose to use the 3rd party service Twilio to handle all the routing for us because it has already been deployed and proven to work. We could take on the task of implementing something like this ourselves, but there really is no point in reinventing the wheel. By delegating this work to Twilio's reliable service, we can primarily focus our attention on what is specific to our project: accepting and coordinating rides for those in need.

## **Heroku**

We chose Heroku to cover our server hosting because a cloud-based server architecture will allow us to easily deploy to Uganda. If we decided to own and manage our own hardware, then we would have to ensure that it is constantly running, buy new hardware if our current system could not handle the traffic, etc. Heroku allows for quick and easy scaling, as well as guaranteed up-time and customer support.

## **Node.JS**

We needed to run some sort of framework on our server so we can accept and respond to HTTP requests incoming from Twilio. Node.JS offers a myriad of "modules" that are easy to install and integrate, including Express, our web framework. Node is also event driven rather than thread based, so incoming requests do not have to queue up while a current request is being handled. It will allow for easy scaling, while the availability of node modules helps speed up development (we can arbitrarily use pre-made modules to cover common challenges that we would otherwise have to spend time working through).

## Chapter 3. Testing

To ensure the integrity of our system, we have formulated a test plan to identify any bugs or logical gaps. We have included our plan below, along with the results and recommendations for future testing.

### 3.1 Test Plan

The logical flows for both drivers and riders were each tested individually for errors. Each test includes pre-conditions, a scenario of the use-case being tested, and the expected results for each step of the use-case.

#### Rider Tests

##### **Requesting Rides**

*Preconditions: None*

<i>Steps</i>	<i>Expected Results</i>
1. Text "RIDE"	1. Receive reply soliciting location
2. Text number corresponding with one of the available locations	2. Receive reply asking if a trailer is required
3. Text back "YES" or "NO"	3a. If a driver is available, receive a confirmation text 3b. If no drivers are available, receive a text about no availability
4. If no driver available, wait for text if driver becomes available	4. If by 30 minutes no drivers are available, receive text saying so. Otherwise, receive text about ride availability.

Table 3.1 Requesting Rides

## Reviewing Drivers

*Preconditions: User has been given a ride*

<i>Steps</i>	<i>Expected Results</i>
1. Driver ends trip (no rider input required)	1. Receive text asking if ride was satisfactory
2a. Text “YES” 2b. Text “NO”	2a. Receive text thanking user 2b. Receive text asking for comments
3. If responded with NO, then send text with any feedback	3. Receive apology text, thank user

Table 3.2 Reviewing Drivers

## Driver Tests

### Changing Driver Availability

<i>Steps</i>	<i>Expected Results</i>
1a. Driver status is unavailable, driver texts “START SHIFT” 1b. Driver status is available, driver texts “END SHIFT”	1a. Receive reply requesting driver location (by region) 1b. System notifies driver of successful request “You have ended your shift. Thank you.”
2a. Driver replies with current location for work, using a number	2a. System notifies driver of successful request “You have started your shift in _____.”

Table 3.3 Changing Driver Availability

### Start/End Rides

*Preconditions: Driver has responded to ride request and picked up rider*

<i>Steps</i>	<i>Expected Results</i>
1. Driver texts “START RIDE”	1. System records start time and responds with acknowledgement
2. Driver texts “END RIDE”	2a. System records end time and calculates total ride time for analytics, reports time to driver

	2b. System returns driver status to “available”
--	---

Table 3.4 Start/End Rides

## Responding to Ride Requests

*Preconditions: Driver is listed as available to work*

<i>Steps</i>	<i>Expected Results</i>
1. Submit ride request in same location as driver	1. Driver receives text “A user in your location has requested a ride. Accept?”
2a. Text “YES” 2b. Text “NO”	2a. Driver receives text containing rider’s phone number for pickup and is listed as “unavailable” 2b. Driver receives text “You have declined a ride.”
3a. Submit another ride request	3a. Driver should not receive request (currently giving ride)

Table 3.5 Responding to Ride Requests

## 3.2 Test Results

Most testing was completed throughout the development process. After significant changes were made, each of these tests was run to ensure that the integrity of the system had not been compromised. If a test did not pass, the changes were modified until each test passed successfully. Additionally, we requested the assistance of friends and family to stress-test our application and servers. The application currently completes the test plan without fail.

## 3.3 Future Testing

Ty will be responsible for maintaining the application in Uganda and for completing future testing to ensure the service works as intended. The application has only been tested in the United States, so general testing must take place in rural Uganda before deployment is possible. Connectivity is the main cause for concern with regard to future testing. Future developers will need to make sure that our toll-free Twilio number is able to handle messages from international users. The server’s ability to handle high volumes of messages will also need to be tested, and the servers should be expanded as necessary. Our current test plan will be sufficient for testing the application’s reliability in Uganda.

## Chapter 4. Societal Issues

Having created a service intended for a rural African environment, we intend on improving the lives of our potential users. We considered the many different societal issues that would have an impact for each design decision. Below is a summary of how Wakabi addresses many of these issues.

### 4.1 Ethical

From the beginning of the project's inception, our purpose has been to empower the local entrepreneurs, drivers, and general public of rural Uganda. We have aimed to minimize any negative impacts that may result from Wakabi's deployment. In the United States, we have seen an uprise from local taxi companies directed towards the extension of on-demand services like Uber and Lyft. These services have proved to be in direct competition with taxi companies, resulting in diminishing income for said companies. We have designed Wakabi in such a way that rather than competing with the current transportation workers of Uganda, we will be helping them. We will not be taking away their customers and income, but rather providing them with more customers and higher income.

### 4.2 Economic

An important aspect that we needed to carefully consider is how to collect income without taking any profit from our Boda drivers. If we followed similar models as used here in the United States, we would take a percentage of each ride cost. If we kept ride prices at the rate currently being charged, that means each driver would only receive a percentage of what they used to make from each ride. Our solution was to allow drivers to keep the total ride fare as they are doing now, but charge them a monthly fee in order to stay registered with Wakabi. This way, we do not cut into their income per ride, but their monthly fee will be justified by the extra profit they are receiving from the increased customer traffic provided by Wakabi.

Another economic improvement that Wakabi provides is the ability to tax revenue from Boda rides. Since each Boda driver currently works independently and directly collects fare from riders, the government has no way of taxing these transactions. However, if we collect monthly fees from our riders, the government will now have a way of taxing the Boda industry. In turn, these tax funds will hopefully be used to improve public utilities,

including the very roads that our Boda drivers rely on. So when considering the ecosystem as a whole, Wakabi will provide more riders for Boda drivers, who will increase the drivers' incomes, who will pay a monthly fee, which will be taxed by the government, who will then improve public utilities.

### 4.3 Health and Safety

By offering a transportation service in a region known to have poor roads, as well as the fact that our drivers use motorcycles, we needed to be especially mindful of the safety of our users, both drivers and riders. By registering our drivers into a single system, as opposed to having independently operating drivers, we can provide them with standard training and maintenance guidance. We can inform them of the safest practices while driving, show them how to maintain their motorcycles to ensure they do not fail during operation, and possibly provide helmets for both the drivers and their passengers. We are accepting the liability of our riders when they choose to use Wakabi, so we must ensure their safety at utmost importance.

### 4.4 Usability

When Ty was conceiving the original idea for Wakabi, one of the most important aspects of the service was that it was intended for the people of Uganda. This means that it would not be a smartphone app, as only a small percentage of Ugandans could utilize it, and that all diction would be presented in their native language, Luganda. We have designed the necessary user interactions to be as short and simple as possible, usually only requiring one or two words from the user for any given response. Thanks to the Twilio service, there is a possibility in the future of expanding Wakabi to not only be accessible via SMS, but also through an automated voice service (meaning users could call the Wakabi number and be guided through the request process by listening to options and indicating selections via the phone number pad). This will allow people that are illiterate to also use our service (an important consideration since not everyone in rural Uganda can read and write).

## Chapter 5. Conclusions

### 5.1 Maintenance Guide

Following the 2014-15 academic year, this project will be handed off to future engineers to continue development and prepare for deployment in Uganda. The separate components of the project have been broken down below along with information concerning future development.

#### **Twilio**

A Twilio number has been created to act as the Wakabi number accessible to the public. Twilio allows the creation of multiple phone numbers per account, so additional numbers could be generated in the future (perhaps for providing information to drivers only).

From the Twilio dashboard, you can specify separate URLs for incoming text messages and incoming voice calls per number. The voice URL is currently set to the default Twilio demo server, but can be changed to a URL on our server in the future.

Each text costs approximately \$0.0075 and is subtracted from the account balance. An option can be enabled to auto-load the balance when it drops below a threshold.

We are currently in the process of setting up a relationship with a Twilio evangelist who will be able to aid with any Twilio related problems or questions.

#### **Heroku**

With Heroku, we pay for the amount and quality of “dynos”. A dyno is Heroku’s unit of computing power, so you can pay for multiple dynos at a certain power. We currently have 1 free dyno, but once traffic increases to the point that exceeds the dyno’s capability, paying for more powerful dynos will help scale.

Heroku’s website has extensive documentation concerning the heroku “toolbelt” (their command line interface) which you can use for deployment, restarting dynos, manage add-ons, etc. The most important feature of their toolbelt is for deployment. There is a git repository hosted on Heroku that holds the current app code. In order to deploy a new version, you simply have to push your changes to that master branch (“git push heroku master”).

## **The code + Node**

We currently have incoming texts routed to `wakabi.herokuapp.com/incoming`. The request is handled by `incoming.js` under the 'routes' folder. We do minimal processing here before passing the request to either `RiderMessenger` or `DriverMessenger` (after we've determined if the sender is a rider or driver).

Since we have to keep track of a conversation (either between us and a rider or between us and a driver), we use cookies with Twilio. With outgoing responses, we set their `rideStage/driveStage` depending on what stage they are in their respective flows. You can view the different stages for riders and drivers in `stages.js`.

The `TextMessenger.js` file handles all outgoing texts. When we have determined the correct response in `RiderMessenger` or `DriverMessenger`, we call either `text` or `textResponse` (depending on whether we're texting in response to a request or sending a 'cold' text) from `TextMessenger.js`.

Most of the database operations have been separated into `db.js`. These queries use an asynchronous, callback method. Most calls to these functions don't require immediate response (because of the asynchronous nature), but the code can be refactored in order to pass a callback to the `db` function to keep the logic within `Rider/DriverMessenger`.

## **Administrator Dashboard**

As previously mentioned, incoming texts get routed to `wakabi.herokuapp.com/incoming`. However, requests to `wakabi.herokuapp.com/` return a web page. These pages are written in Jade (which compiles to HTML) and styled by LESS files (which compile to CSS).

The homepage shows a dashboard with driver and rider stats of the current day, as well as a column showing alerts (which for now just entail driver ratings dropping below an accepted threshold).

The site will also have capabilities to add and remove drivers to the system through a simple form.



## 5.2 Suggested Changes

Wakabi is in a fully-functional state, ideally ready for testing in Uganda. Ty Van Herweg will take the application to Uganda on a Fulbright Scholarship and continue its deployment. Certain changes should be considered to ensure the application's stability and success. Some of the suggested changes are mandatory, while others may be implemented as the developers see fit.

1. All user-facing text must be translated to Luganda
  - a. Strings have been separated into their own file to make this process easier
2. Need to implement a payment system to collect the monthly fee from drivers
  - a. Determine pricings for trailers and rides (e.g. distance traveled, time, or by zone)
3. Changing the Twilio number to a Ugandan country code might be beneficial
  - a. Ideally, we would want to keep the number "Toll-Free" to minimize user costs
  - b. See Twilio's newly announced "Messaging Copilot" that offers "geo-match"
4. Increase the number of dynos on the Heroku application server as needed
5. Formal driver sign-up process once in Uganda
  - a. An administration page will be provided to add drivers to the database
6. Create additional database views to capture important data
  - a. For example, show driver ratings or create "driver/rider profiles"
7. Consider investing in analytics software to monitor application performance metrics
8. Explore voice-response services provided by Twilio to increase usability
9. Integrate the application with mobile-banking services, or other SMS-based applications
10. Try to minimize the amount of database queries made by the server
  - a. Look for ways to maintain short-term data
11. Storing phone numbers in the database could provide marketing opportunities or the ability to share important safety information to drivers/riders
  - a. For example: road hazards, weather conditions, gas prices

### 5.3 Lessons Learned

Creating Wakabi has been an insightful, exciting, and rewarding experience for all members involved. It was motivating to know that our project would eventually be deployed to serve an actual need in rural Uganda. Working on a long-term project has been a daunting task, but the senior design project provided us the opportunity to solve a real-world problem from start to finish. We were exposed to several different processes that businesses execute when working on a project, such as creating a design report and even presenting that design report in front of a group of important people. We also experienced the stresses and frustrations that accompany developing a large project that integrates with many different types of software. In the end, we became familiar with the formal process of developing a project – an experience that will prove vital to our success as computer engineers.

One important lesson we learned was to maintain flexibility when researching technologies during the early stages of project development. Our initial design report contained detailed information about technologies we thought we would be using (Tropo, RapidSMS, Python). However after performing additional research, we found that our application would likely perform better if we switched architectures to Twilio and Node.JS. Making the switch meant that we needed to spend time updating our reports and learning unfamiliar technologies, but our flexibility allowed us to build an even more robust application than we had originally thought possible.

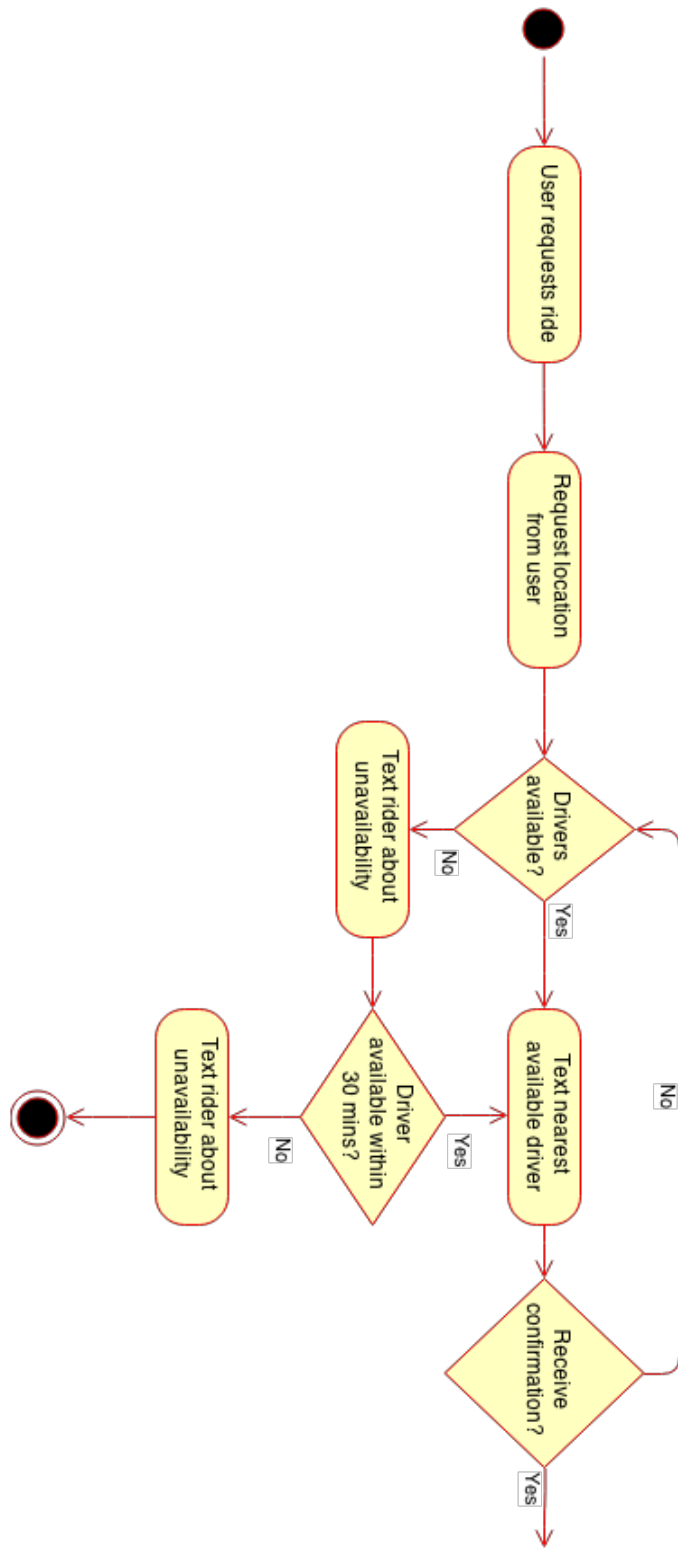
Another lesson we learned while developing Wakabi was how to effectively manage several deadlines while simultaneously dividing work between the two of us, and knowing that the work would get done. Maintaining constant trust and respect between teammates meant that we could focus less on management and more on development. It also alleviated some of the stresses that normally accompany group projects, such as worrying about the quality of the other person's work. In the work force projects will almost always be completed in teams, so it was important that we learn how to cooperate with other people as effectively as possible.

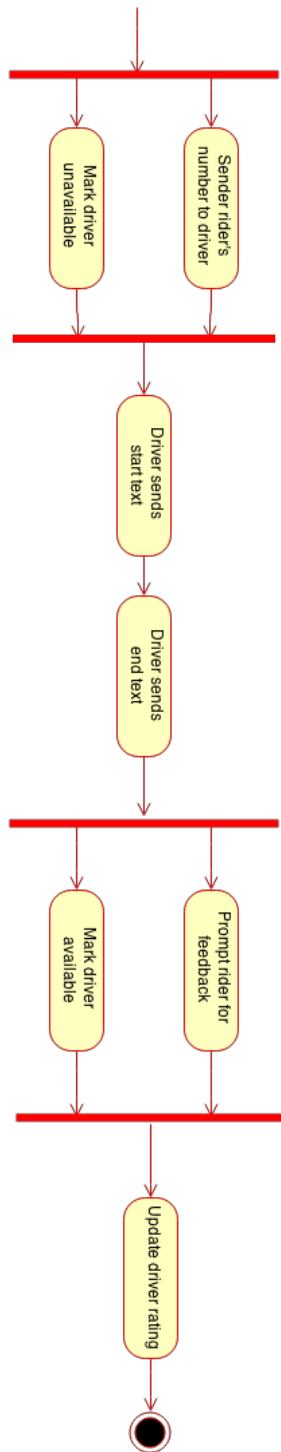
## Chapter 6. References

1. "Rural Population." The World Bank. International Development Association, 2014. Web. 07 June 2015.
2. Kalba, Kas. Africa's Mobile Money Story. *Intermedia*, 41 (5), 26- 29.
3. Siedner, M.J., Haberer, J.E., Bosco Bwana, M., Ware, N.C., and Bangsberg, D.R. High acceptability for cell phone text messages to improve communication of laboratory results with HIV-infected patients in rural Uganda: a cross-sectional survey study. *BMC Medical Informatics & Decision Making*, 12 (1). 56-62.
4. Kisaalita, W.S. and Sentongo-Kibalama, J. Delivery of urban transport in developing countries: the case for the motorcycle taxi service (boda-boda) operators of Kampala. *Development Southern Africa*, 24 (2). 345-357.

# Chapter 7. Appendices

Appendix A - Activity Diagram:

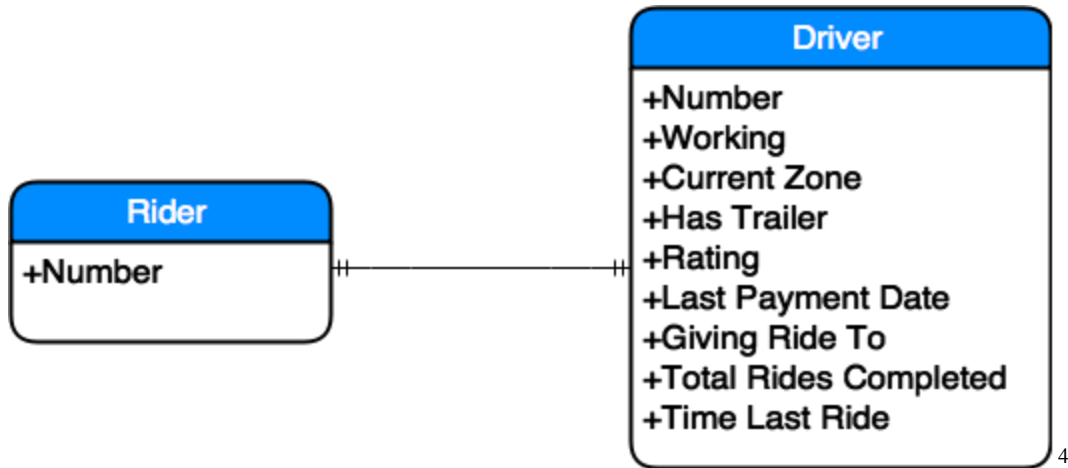




3

<sup>3</sup> Figure 7.1 Activity diagram describing system processes.

*Appendix B - Entity-Relationship Diagram:*



**Rider**

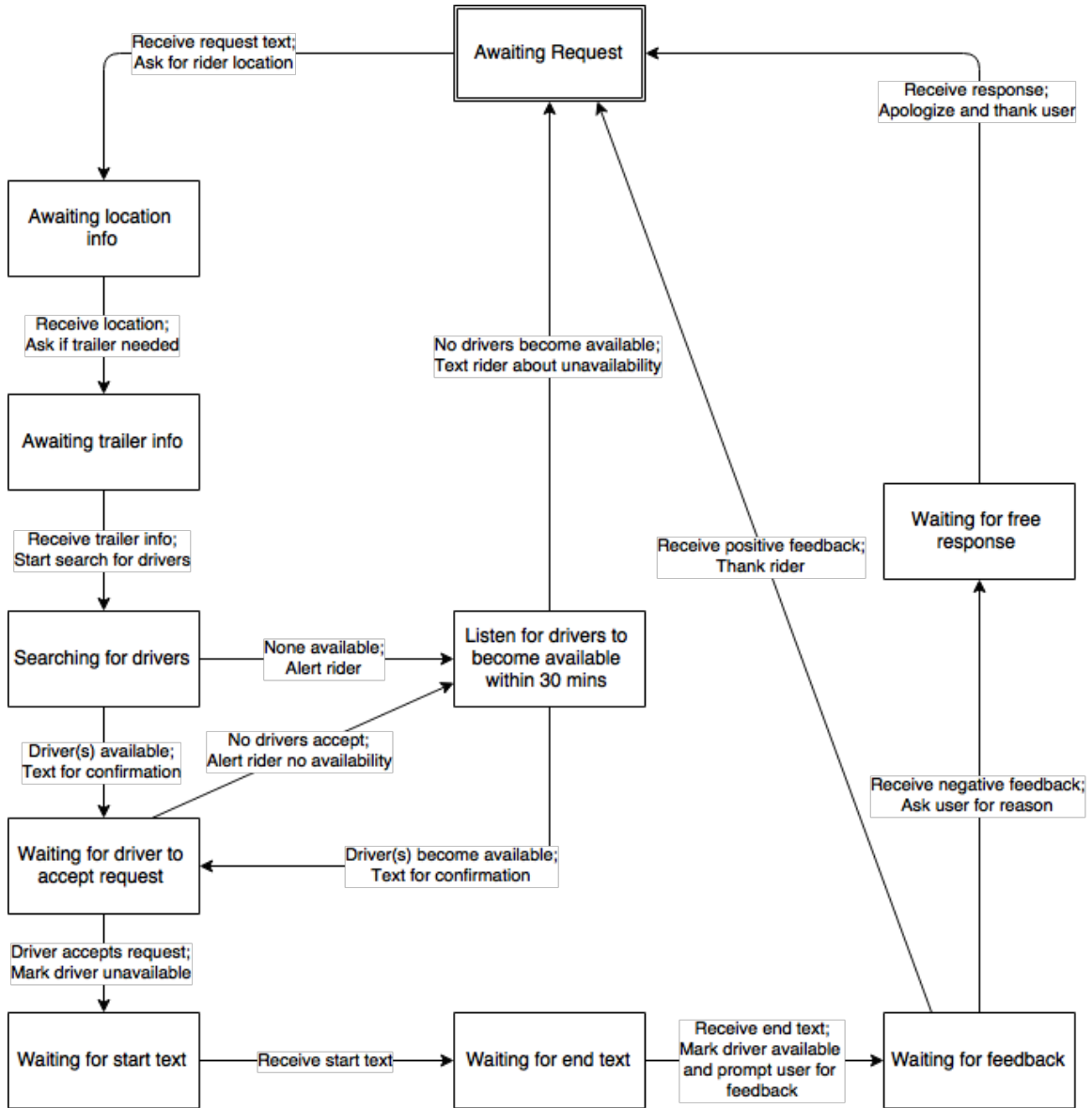
- Number: Rider’s phone number

**Driver**

- Number: Driver’s phone number
- Working: Boolean indicating if driver is “on shift” and available to take new ride requests
- Current Zone: The zone the driver is currently in
- Has Trailer: Boolean indicating if driver has a trailer
- Rating: Driver’s rating on a scale from 0% to 100%
- Last Payment Date: The date of their last payment. If their last payment is not within the past month, they will not receive new requests.
- Giving Ride To: If they are currently giving a ride, this field holds the rider’s phone number
- Total Rides Completed: The total number of rides the driver has successively started and ended
- Time Last Ride: The time at which they ended their latest ride. Used to break ties when receiving a request in a zone with multiple drivers (request first sent to driver that has waited longest).

<sup>4</sup> Figure 7.2 Entity-Relationship diagram describing relevant information about entities and their relationships.

Appendix C - Component State Diagram:



5

<sup>5</sup> Figure 7.3 Component State diagram describing various system states and subsequent actions.