

Philly Bike Report: A Mobile App for Mapping and Sharing Real-Time Reports of Illegally
Blocked Bike Lanes in Philadelphia

by

Douglas Montgomery

A Thesis Presented to the
Faculty of the USC Graduate School
University of Southern California
In Partial Fulfillment of the
Requirements for the Degree
Master of Science
(Geographic Information Science and Technology)

August 2017

Table of Contents

List of Figures	vii
List of Tables	x
Acknowledgements	xi
List of Abbreviations	xii
Abstract	xiv
Chapter 1 Introduction	1
1.1. Project Study Area	2
1.2. Cycling Infrastructure	3
1.2.1. Definitions.....	3
1.2.2. Cycling Network of Philadelphia.....	5
1.3. Motivation.....	7
1.3.1. Recent Trends Towards Increased Cycling	7
1.3.2. Cyclist Safety in Philadelphia.....	8
1.3.3. Frequency of Blocked Bike Lanes.....	9
1.3.4. Encourage Focus on Protected Bike Lanes.....	10
1.3.5. Facilitate Enforcement of Bike Lane Restrictions	11
1.3.6. Support Advocacy Efforts of the Local Cycling Community	12
1.3.7. Benefits of Cycling as Transportation	13
Chapter 2 Related Work.....	15
2.1. Volunteered Geographic Information	15
2.1.1. VGI Definition.....	15
2.1.2. VGI Data Quality and Cycling	16

2.2. Existing Mobile Cycling Apps.....	19
2.2.1. BCApp	19
2.2.2. Cyclopath	19
2.2.3. BikeMaps	20
2.2.4. MyBikeLane	22
2.2.5. Other Existing Cycling Apps	25
Chapter 3 Methodology	27
3.1. Project Objectives	28
3.2. Intended User Groups	28
3.3. User Requirements.....	30
3.3.1 Locate User	30
3.3.2. Exchange Information About Blocked Bike Lanes	30
3.3.3. Tweet to #UnblockBikeLanes.....	31
3.4. Database Selection.....	31
3.4.1.Comparison of SQL and NoSQL Databases.....	32
3.4.2. NoSQL Social Media Benefits.....	36
3.4.3. NoSQL Database Selection Criteria	36
3.4.4. Parse Server Overview.....	38
Chapter 4 Programming And Design.....	41
4.1. Programming Languages and Technologies	41
4.1.1. Development Environment	41
4.1.2. Google Maps Android API	42
4.2. App Programming.....	43

4.2.1. App Organization.....	43
4.2.2. User Management.....	45
4.2.2. Determine User Location.....	48
4.2.3. Display Incidents from Database.....	50
4.2.4. Submit Incidents to Database.....	52
4.2.5. Display Recent Tweets.....	53
4.2.6. Tweet to #UnblockBikeLanes.....	54
4.3. Database Structure and Design.....	54
4.3.1. Connecting to the Database.....	54
Chapter 5 Results.....	56
5.1. App Organization.....	56
5.2. Main Map.....	58
5.3. Incident Info Windows.....	59
5.4. Report Incident Fragment.....	60
5.5. Navigation Menu and Other Activities.....	62
Chapter 6 Conclusions.....	65
6.1. Key Findings.....	65
6.2. Reference Resources.....	66
6.3. Future Work.....	66
6.3.1. Further Twitter Integration.....	66
6.3.2. Additional Data Layers.....	67
6.3.3. iOS and Web Support.....	68
6.3.4. Data Sharing.....	69

6.3.5. Next Steps	69
References.....	71

List of Figures

Figure 1. Project study area (Wikimedia Commons).....	2
Figure 2. Bike lane types (FHWA, 2015).....	4
Figure 3. Philadelphia's Cycling Network	6
Figure 4. Collisions involving bicycles since 2014 (NBC10 News Philadelphia)	8
Figure 5. Heat map of blocked bike lane tickets issued between	12
Figure 6. Smartphone Ownership by Income and Age (Pew Research Center's Internet & American Life Project, 2013)	18
Figure 7. <i>BikeMaps</i> Web Version Homepage.....	21
Figure 8. <i>BikeMaps</i> VGI Collection	22
Figure 9. <i>MyBikeLane</i> Mobile App	23
Figure 10. <i>MyBikeLane</i> Mobile App	24
Figure 11. <i>MyBikeLane</i> “Repeat Offenders”	24
Figure 12. <i>Bike2Go</i> App.....	26
Figure 13. <i>Social Cyclist</i> Mobile App.....	26
Figure 14. Comparison of RDBMS and NoSQL Databases (Hashem and Ranc 2016).....	33
Figure 15. Read Time Comparison for SQL and NoSQL (Maia, Mendonça, Camargos, Holanda, and Maristela, 2016)	34
Figure 16. Write Time Comparison for SQL and NoSQL (Maia, Mendonça, Camargos, Holanda, and Maristela, 2016)	35
Figure 17. Comparison of NoSQL Database Services (Batschinski 2016)	38
Figure 18. Hosted Parse and Parse Server Comparison (Parse 2016)	39
Figure 19. XML code to determine map layout.....	44

Figure 20. Code to determine whether PBR has previously been launched on device	45
Figure 21. Code used to register new user	46
Figure 22. Code used for user login.....	47
Figure 23. User management in Parse Dashboard	48
Figure 24. Code used to locate user	48
Figure 25. Code used to determine Location Services permissions	49
Figure 26. Code used to determine new location when user moves.....	50
Figure 27. Code used call GoogleAPIClient.....	50
Figure 28. Code used to pull incidents from the database	51
Figure 29. Code used to place incident markers on map	51
Figure 30. Code used to create new object in database	52
Figure 31. Code used to update object with additional information.....	53
Figure 32. Code used to query tweets.....	53
Figure 33. Code used to compose tweets.....	54
Figure 34. Code used to connect to Parse Server and Back4App.....	55
Figure 35. Code used to import Parse libraries.....	55
Figure 36. App Organizational Flowchart	57
Figure 37. App welcome screen.....	58
Figure 38. Main Map Screen	59
Figure 39. Incident Window A.....	60
Figure 40. Incident Window B.....	60
Figure 41. Report Incident Fragment.....	61
Figure 42. Twitter Dialog Prompt.....	62

Figure 43. Twitter Activity	63
Figure 44. About Activity.....	64
Figure 45. Statistics Activity.....	64

List of Tables

Table 1. User Requirements.....	30
Table 2. Potential Future Datasets.....	68

Acknowledgements

I am immensely grateful to my thesis chair, Dr. Elisabeth Sedano, for providing invaluable guidance and support throughout the duration of this project. I am also very thankful to my thesis committee members, Dr. Yao-Yi Chiang and Dr. Darren Ruddell, for their assistance and insight. I would also like to thank the Philadelphia cycling community, particularly Twitter advocates and activists, for providing the inspiration to create this app. Last but not least, I would like to thank my family and friends, especially my parents, for the endless support, encouragement, and love they have shown me throughout this entire process.

List of Abbreviations

ADT	Android Developer Tools
API	Application Programming Interface
AVD	Android Virtual Device
AWS	Amazon Web Services
BaaS	Backend as a Service
BCGP	Bicycle Coalition of Greater Philadelphia
DBMS	Database Management System
DOT	Department of Transportation
XML	Extensible Markup Language
FHWA	Federal Highway Administration
GIS	Geographic Information Systems/Science
GIST	Geographic Information Science and Technology
GPS	Global Positioning System
IDE	Integrated Development Environment
JAR	Java Archive
NoSQL	Not Only SQL
OSM	Open Street Map
PBR	Philly Bike Report
PGIS	Participatory Geographic Information System
PPA	Philadelphia Parking Authority
PUMA	Public Use Microdata Areas
RDBMS	Relational Database Management System

SDK	Software Development Kit
SQL	Structured Query Language
UI	User Interface
UX	User Experience
VGI	Volunteered Geographic Information
VGIS	Voluntary Geographic Information System

Abstract

Cycling as a form of urban transportation has been growing in popularity across the United States over the past several years. While many cities have added protected bike lanes in recent years, the City of Philadelphia does not have a single protected bike lane, despite the fact that among American cities with one million or more people, Philadelphia has the highest share of commuters who bike to work. Due to the unprotected nature of cycling infrastructure in Philadelphia, bike lanes are routinely blocked by motor vehicles, presenting a significant safety hazard to cyclists. Previous efforts to raise awareness of blocked bike lanes – including a campaign by the Philadelphia Parking Authority encouraging cyclists to tweet the location and photographic evidence of blocked lanes to the #UnblockBikeLanes Twitter hashtag – have been ineffective. Therefore, this project aims to create a more robust method for documentation of blocked bike lanes in Philadelphia, through use of an Android app that provides a spatial representation of blocked bike lane occurrences. The app, named Philly Bike Report (PBR), utilizes a cloud database to allow users to view and report recently blocked bike lanes. In addition to the core focus on collection and display of volunteered geographic information on cycling conditions, PBR also allows users to contribute to the #UnblockBikeLanes Twitter campaign by providing the option to tweet the incident upon submission. By creating a mobile app and accompanying cloud database of blocked bike lanes, PBR aims to provide a more effective method for viewing and reporting blocked bike lanes in Philadelphia. The key findings of this thesis are represented by the creation of PBR, as a demonstration of how a mobile app with a cloud database can be used to view and report blocked bike lanes.

Chapter 1 Introduction

Cycling as a form of urban transportation has been growing in popularity across the United States over the past several years. Many cities have increased their focus on cycling infrastructure, in an effort to improve cyclist safety and promote continued increases in cycling. While many cities have been adding protected bike lanes in recent years, the City of Philadelphia does not have a single protected bike lane, even though Philadelphia has the highest share of commuters who bike to work among American cities with one million or more people (Beck 2015). Unsurprisingly, this combination of high bicycle ridership and lack of protected bike lanes has led to dissatisfaction in the Philadelphia cycling community, as bike lanes are routinely blocked by motor vehicles, resulting in unsafe conditions.

In order to effectively advocate for improved cycling infrastructure, it is important to collect data on current challenges affecting the safety and convenience of traveling by bike in Philadelphia. Therefore, the central objective of this project is to create a mobile GIS application, named Philly Bike Report (PBR), with which users can report blocked bike lanes that hamper their ability to safely and efficiently travel by bicycle in the City of Philadelphia. Users of the app are able to submit information about the incident as well as photographic evidence. Once the incident is submitted, PBR updates in real time with the new information. In addition to collection and display of volunteered geographic information (VGI) on blocked bike lanes, PBR allows users to view and contribute to the Philadelphia cycling community on Twitter using the #UnblockBikeLanes hashtag.

1.1. Project Study Area

The study area for this project is the City of Philadelphia, and users are able to report incidents anywhere within the city limits. The study area is shown in Figure 1.

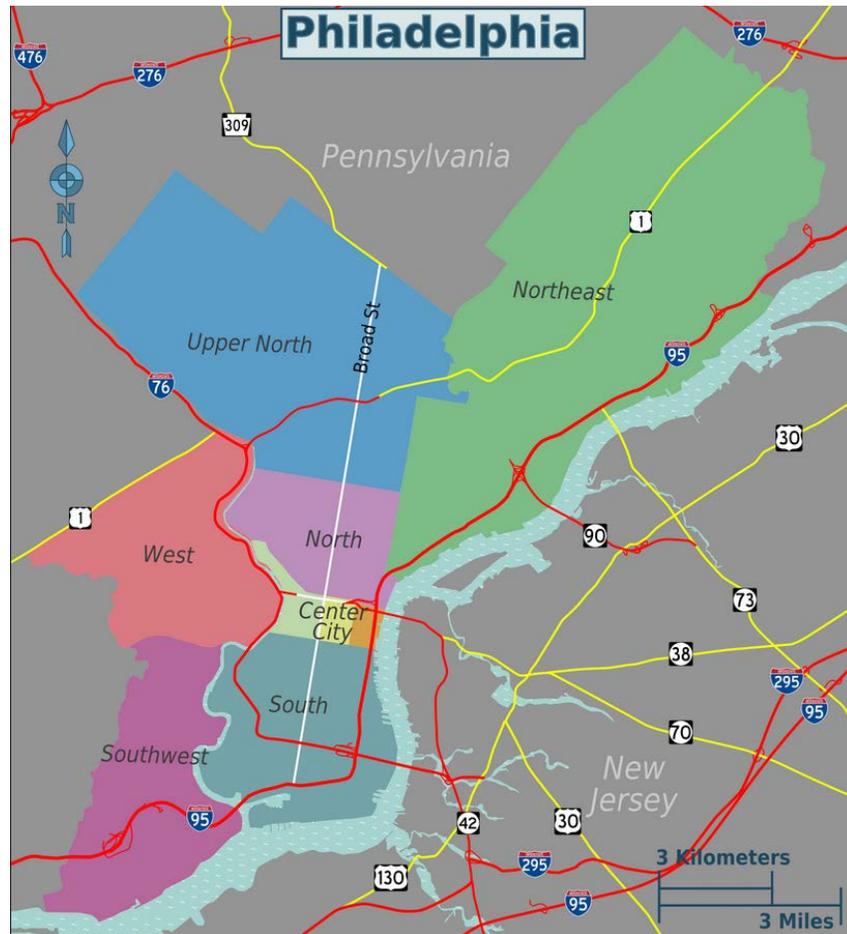


Figure 1. Project study area (Wikimedia Commons)

The popularity of cycling in Philadelphia can largely be attributed to the history of the city's development, as well as its topography. First, Philadelphia is a relatively old city by American standards, and much of the city – especially the Center City, where cycling is most prevalent – was developed prior to the advent of automobiles. As such, Philadelphia is quite dense, and thus conducive to travelling by bike. Second, the topography of Philadelphia is rather flat, which is also beneficial to cycling. The positive impact of Philadelphia's density and topography is echoed by The Center City District and Central Philadelphia Development

Corporation's report on bicycle commuting: "Philadelphia's dense, compact live-work downtown and relatively flat topography are excellent for bicycle commuting, especially for the 42% of working residents who live and work in Greater Center City" (2016, 2).

1.2. Cycling Infrastructure

Cycling infrastructure comes in many different forms, and there are important differences between bike lane types. Some types of bike lanes are far less likely to be blocked by motor vehicles than others, and many cycling advocates argue that such lanes are inherently safer and more efficient. This section outlines various bike lane definitions, followed by an examination of which types are most prevalent in Philadelphia and how that relates to this project.

1.2.1. Definitions

The United States Department of Transportation (DOT)'s Federal Highway Administration (FHWA) divides bike lanes into six main categories, listed by degree of separation from motor vehicle traffic from least to most: signed routes, shared lane markings, on-street bike lanes, on-street buffered bike lanes, separated bike lanes (also called protected bike lanes and cycle tracks), and off street trails (2015). These types are defined in Figure 2.



Figure 2. Bike lane types (FHWA, 2015)

As can be seen in the FHWA definitions, there is a significant range of bike lanes types, and separated bike lanes offer much greater levels of protection from motor vehicles. *People for Bikes* – a cycling advocacy group – defines protected bike lanes similarly to FHWA. Their definition describes three key characteristics of protected lanes: (1) physical separation between motor vehicle and bicycle traffic, typically in the form of curbs, planters, plastic posts, bollards,

or parked cars; (2) designated as only for use by people on bikes; and (3) located on or adjacent to the main street grid (2014).

The presence of physical separation is key to improved cyclist safety, as such lanes require less direct interaction between people on bikes and people driving motor vehicles. In his evaluation of the relative protection level provided by different types of bike lanes, Andersen (2014) provides support for the benefits of protected bike lanes. Andersen analyzes fourteen types of bike lanes – one striped (unprotected) buffered bike lane and thirteen different protected bike lane styles – and assigns a protection score on a scale of one to five for each type. The buffered bike lane receives a score of two out of five, while the protected bike lane scores range from three out of five for lanes protected by low bumps, to five out of five for lanes protected by curbs, bollards, planters, or large bumps. Buffered bike lanes are inherently less safe than protected bike lanes, as cars and trucks can enter buffered lanes but cannot enter protected lanes, due to the presence of a physical separation between people on bikes and motor vehicle traffic.

1.2.2. Cycling Network of Philadelphia

According to the Philadelphia Streets Department (2015), the city has 229.4 miles of conventional bike lanes, 18.2 miles of on-street buffered lanes, and 36.7 miles of shared lane markings, but does not have any protected bike lanes. The Philadelphia cycling network is shown in Figure 3. The screenshot is from the web version of PBR, which was created prior to the Android version discussed in this thesis. The web version – which is discussed in greater detail in Chapter 6 – is a rather basic demonstration project, and it does not currently utilize the same database as the Android version.

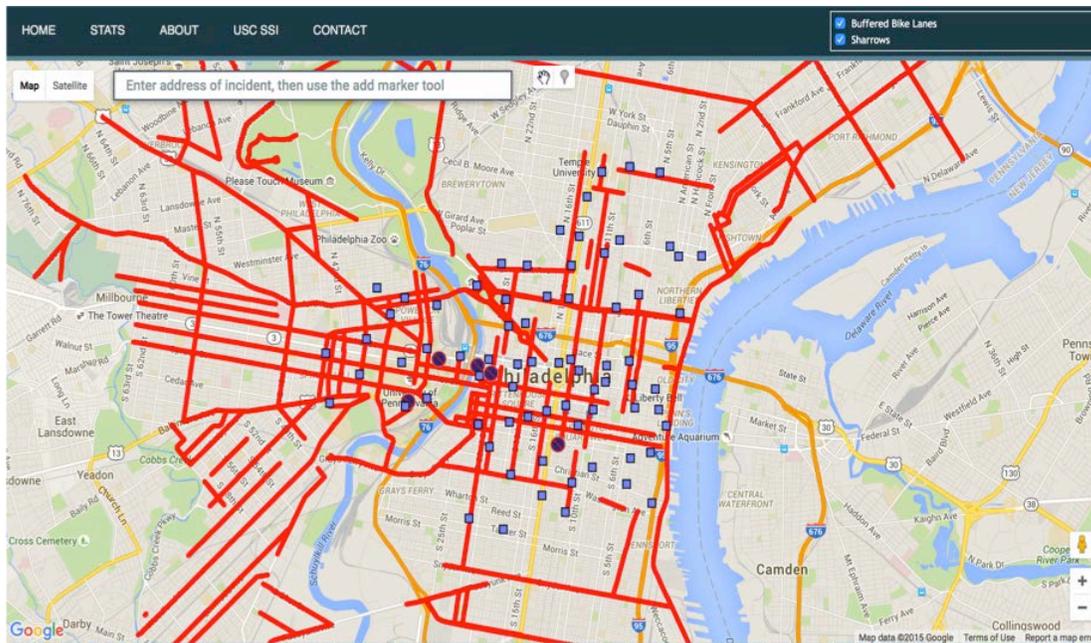


Figure 3. Philadelphia's Cycling Network

It might seem counterintuitive that Philadelphia has relatively high cycling rates, given the city's lack of protected bike lanes. This reality of high ridership despite rather basic cycling infrastructure suggests that installation of protected bike lanes would likely lead to further increases in cycling rates. Evidence from the National Association of City Transportation Officials (NACTO) supports the notion that improved cycling infrastructure leads to increased ridership, as their 2016 study of cities throughout North America found that the addition of protected bike lanes led to increased cycling rates on those streets by 21% to 171%.

In addition to bike lanes, Philadelphia also has a bike share system managed by the company Indego. The system provides bikes available for rent by the hour at stations throughout the city. Monthly memberships are also available. The locations of Indego stations are displayed as blue squares in Figure 3. The success of Indego provides further evidence that efforts to encourage cycling are likely to be successful. Indego began operation in April 2015 and quickly became successful, with 8,300 memberships and 421,000 trips taken by the end of 2015

(Tannenwald 2015). The success of the program has continued, as Indego celebrated its 1 millionth ride on November 10, 2016 (Romero 2016).

1.3. Motivation

There are multiple factors that motivate this project, ranging from combating blocked bike lanes to contributing to the Philadelphia local cycling advocacy community. Each of these factors is discussed in greater detail in the following sections.

1.3.1. Recent Trends Towards Increased Cycling

The popularity of cycling in Philadelphia has grown rather quickly in the past decade, and the trend shows no signs of slowing down. This increase in popularity reflects larger national trends, in which the number of people nationwide who commute by bike increased 60.8 percent between the 2000 and 2008-2012 editions of the American Community Survey (McKenzie 2014). While such strong nationwide growth in biking to work is notable and encouraging to cycling advocates, the increase is even more noteworthy in Philadelphia, where bicycle commuting increased an impressive 260 percent between 2005 and 2013 (Bicycle Coalition of Greater Philadelphia [BCGP] 2014).

The popularity of cycling in Philadelphia is particularly pronounced in the neighborhoods of Center City and South Philadelphia, as defined by the U.S. Census Bureau's Public Use Microdata Areas (PUMAs). South Philadelphia and Center City both have a relatively high bicycle commute mode share, at 5.5 percent and 5.3 percent, respectively. These two neighborhoods rank in the top 25 PUMAs nationwide for cycling to work, and Philadelphia is one of only four cities – along with Portland, Oregon, and San Francisco – to have two PUMAs in the top 25 (BCGP 2014).

1.3.2. Cyclist Safety in Philadelphia

As cycling rates increase, safety concerns continue to worsen, as shown by the fact that since 2015, there have been eleven victims of fatal crashes involving cyclists in Philadelphia, which is much higher than the city's recent historical average of two to four fatal bicycle crashes per year (Spencer and Chang 2016).

A map of bicycle-related collisions reported to the Philadelphia Police Department since 2014 can be seen in Figure 4. Analysis of this data provides further evidence of the need for protected bike lanes, as many of the reported crashes occurred in bike lanes. Indeed, Spruce Street had 63 bicycle-related crashes, despite the fact that it has a painted bike lane, a type of buffered bike lane (Spencer and Chang 2016).

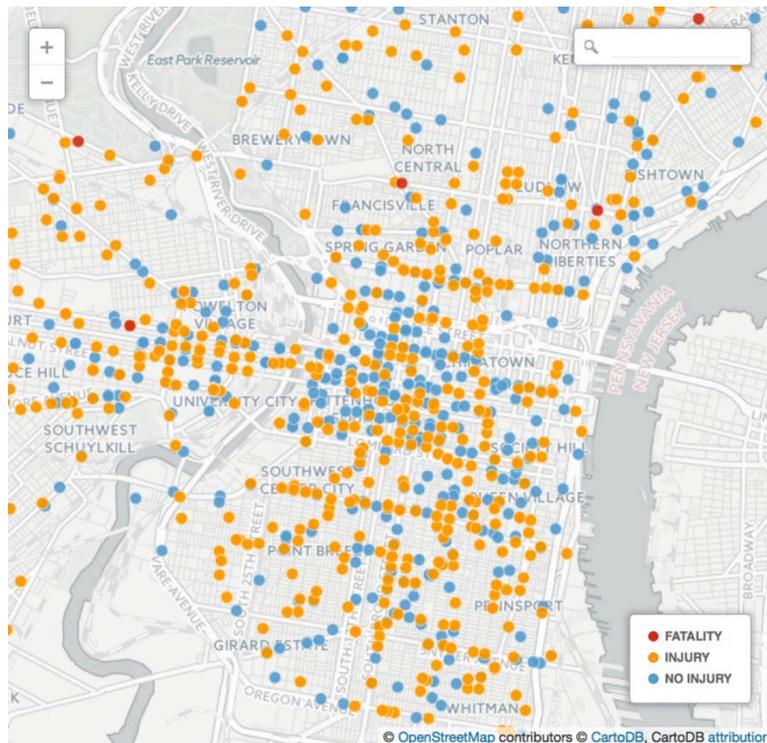


Figure 4. Collisions involving bicycles since 2014 (NBC10 News Philadelphia)

1.3.3. Frequency of Blocked Bike Lanes

The issue of blocked lanes has become so common in Philadelphia that a popular Twitter hashtag – #UnblockBikeLanes – was established in December 2013, as part of a joint effort between the Philadelphia Parking Authority (PPA) and the BCGP. The hashtag is frequently used by Philadelphia cyclists; for example, 53 blocked bike lane incidents were reported to #UnblockBikeLanes in September 2016 alone. While creation of the UnblockBikeLanes hashtag was a useful first step in raising awareness of blocked bike lanes, there has been no method to spatially represent where these instances have occurred in real-time. The frustration of Philadelphia cyclists continues to mount, as the challenge of blocked bike lanes persists, shown by the continued frequent usage of the UnblockBikeLanes hashtag. Many of the tweets also contain accounts of hostile interactions between people on bikes and people blocking bike lanes with their cars, such as drivers honking at cyclists who are legally and appropriately using a bike lane. People who travel frequently by bicycle or car in Philadelphia are likely to experience the high degree of hostility that exists between bikers and drivers in the area. Tweets that recount such hostility provide a powerful picture of the polarized atmosphere of Philadelphia city streets.

Additionally, based on anecdotal evidence from the content of #UnblockBikeLanes tweets, many vehicles have been observed to routinely block a bike lane in the same location each day, without any repercussions. By creating a central database of blocked lanes, this project will allow users to record and track repeat violators. For example, a PBR user can include the license plate number of offending vehicles in their submission, which will then be tracked in the system.

1.3.4. Encourage Focus on Protected Bike Lanes

As city government agencies continue to develop and refine their official plans for improved cycling infrastructure, it is crucial that they provide significant support for the installation of protected bike lanes. Considering that a protected bike lane means there is actual physical separation (such as with bollards or planters) between motor vehicles and space designated for bicycles, it is no surprise that riding in a protected bike lane is a much more safe, efficient, and enjoyable experience than in unprotected bike lanes. Unsurprisingly, cycling statistics reflect the reality that people are more likely to bike when the streets they are travelling contain improved cycling infrastructure, as a 2013 Philadelphia cyclist count conducted by the BCGP found that streets with buffered bike lanes transport 78 percent more cyclists than streets with on-street bike lanes and 131 percent more than streets with no bike lane at all (BCGP 2014).

Not only do protected bike lanes result in a much better experience for people on bikes, they also promote benefits to all Philadelphians, regardless of their preferred method of transportation. This is due to the fact that protected bike lanes provide physical separation between bicycles and motor vehicles, resulting in fewer opportunities for conflicts and a more clear understanding of the rights of a particular road. This notion that protected bike lanes promote certain side benefits is supported by the BCGP cyclist count discussed above, as their work also found that 20 percent of Philadelphia cyclists ride on the sidewalk when no bike lane is present, compared to 8 percent when a standard bike lane is present, and 3 percent when a buffered lane is present (BCGP 2014). Considering that Philadelphia does not have a single protected bike lane, the BCGP's count could not include such lanes in their study. However, the pronounced difference in sidewalk riding with no bike lane compared with a buffered bike lane shows the powerful effect that improved infrastructure can have on cyclist behavior, and it is highly likely that sidewalk riding would be even lower with a protected bike lane (BCGP 2014).

1.3.5. Facilitate Enforcement of Bike Lane Restrictions

As mentioned above, the PPA was instrumental in establishing the #UnblockBikeLanes campaign to report blocked lanes. Considering that the PPA is a city government agency, its support of efforts to document blocked bike lanes shows that the city is willing to support cycling infrastructure and is eager to gain a more complete understanding of the issue of blocked lanes. Indeed, as explained by PPA Senior Administrator Sue Cornell, “We heard what the BCGP had to say and we want to know where these problem spots are so we can know how to best serve the city” (PPA 2013, 1).

The #UnblockBikeLanes campaign was started over three years ago, and the PPA continues to promote the effort. In May 2015, the PPA issued a progress report on the campaign, including summary information on #UnblockBikeLanes tweets and discussion of the campaign’s future. The report explains that a little more than a year after the campaign began, over 270 blocked bike lanes had been reported via the UnblockBikeLanes hashtag (PPA 2015). Also included in the progress report was a heat map of problem areas based on hashtag data collected from December 18, 2013, to December 18, 2014, as shown in Figure 5. The report concludes with a strong affirmation of the PPA’s commitment to combating blocked bike lanes by stating, “...it’s our hope that local residents and visitors to our great city will continue utilizing #UnblockBikeLanes and let us know of any bike lane violations.”

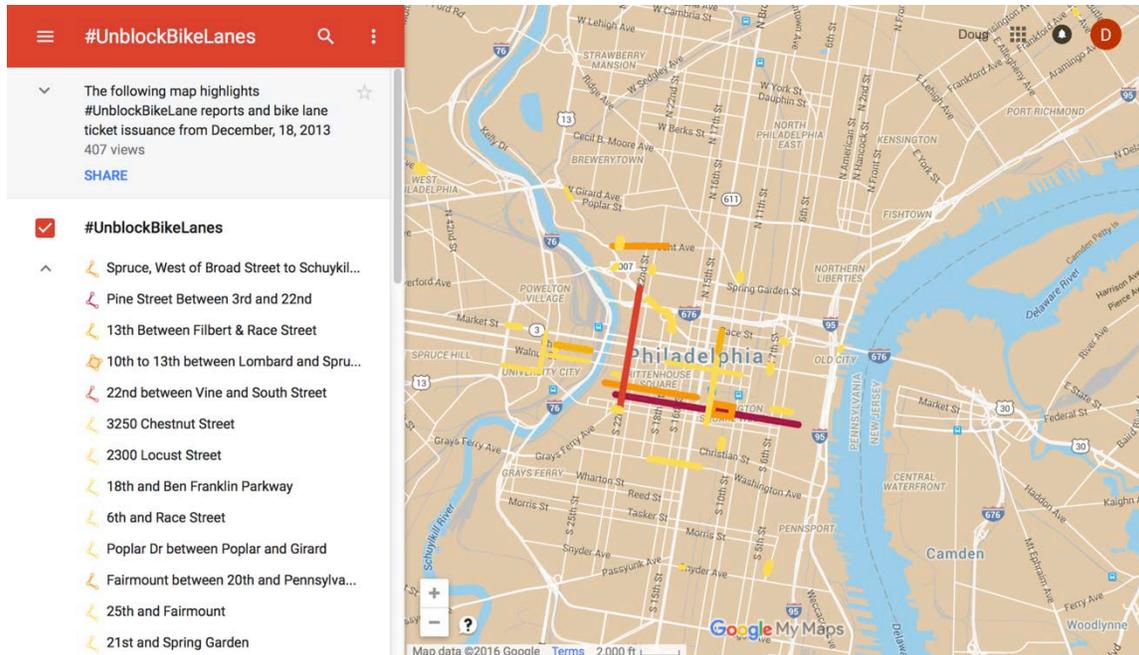


Figure 5. Heat map of blocked bike lane tickets issued between December 18, 2013 and December 18, 2014

1.3.6. Support Advocacy Efforts of the Local Cycling Community

Philadelphia's need for improved cycling infrastructure has been a growing point of contention among the city's cycling community, as cycling in the area is often inefficient and dangerous, despite the fact that a relatively high proportion of Philadelphians bike as their main form of transportation. Although city government agencies in recent years have started to recognize the need for improved cycling infrastructure, efforts to do so have not been aggressive enough for many Philadelphians.

Support of the local cycling community's advocacy is a valuable endeavor, as improved cycling infrastructure can effect real change. Indeed, the initial steps that Philadelphia has taken to improve its cycling infrastructure have already resulted in many tangible benefits. In addition to the increased popularity of cycling discussed above, there have also been measurable improvements in cyclist behavior. Between 2005 and 2013, sidewalk riding by Philadelphia cyclists has decreased by 80 percent, wrong-way riding has decreased by 63 percent, and helmet

use has increased by 95 percent (BCGP 2014). The fact that these indicators have improved so appreciably despite the lack of protected lanes provides further evidence of the trend towards increased cycling popularity.

Providing support for protected bike lanes is a very timely issue, as proof continues to mount that such lanes are needed. In fact, as this thesis was being finalized in December 2016, a new round of community meetings were underway in the Washington West neighborhood of Center City Philadelphia to discuss the possibility of converting the buffered bike lanes on Spruce and Pine Street into fully protected bike lanes. The first of these meetings occurred on November 22, 2016 with a second meeting on December 13. Despite much support for protected bike lanes, there are also strong opinions on the other side of the issue, as evidenced by reports of anti-protected bike lane flyer campaigns that occurred in the neighborhood in advance of the meeting (Lobasso 2016). The presence of this opposition to protected bike lane further shows the need for advocacy in favor of bike lanes.

The importance of involving all segments of the community is highlighted by the diverse demographics of people who bike to work. According to U.S. Census nationwide statistics, people with a graduate degree or above have the highest rates of biking to work, at 0.9 percent, while people at the other end of the education spectrum – people with less than a high school diploma – have the second highest rate, at 0.7 percent (McKenzie 2014).

1.3.7. Benefits of Cycling as Transportation

There are many recognized benefits of cycling as a form of transportation, ranging from enhanced street vibrancy and social cohesion to improved health, lower pollution, and reduced reliance and expenditure on fossil fuels. A growing number of cities around the world have recognized the many inherent benefits of a complete streets approach to planning and have acted

accordingly by implementing improvements focused on cycling, transit, and walking. Studies of cities that have made strong efforts to improve the cycling infrastructure provide evidence of the benefits that increased cycling can provide. For example, Dutch people who cycle regularly have a life span that average six months longer than those who do not cycle (Fishman et al. 2015). The same study also finds that investments in cycling infrastructure result in a high cost-benefit ratio in the long term, as financial savings from the health benefits associated with regular cycling correspond to more than three percent of the Dutch Gross Domestic Product.

Chapter 2 Related Work

This chapter provides a literature review of work related to this project and centers on two main topics: (1) VGI and its use in geospatial analysis; and (2) existing mobile cycling apps.

2.1. Volunteered Geographic Information

At its core, this project relies on VGI to perform its objectives. As such, the app design draws upon existing web and mobile GIS applications that are also based on VGI. When collecting VGI, it is critical to identify the target audience and to understand their motivation for participating. In other words, it is important to understand the *who* and *why* of the app's targeted users.

2.1.1. VGI Definition

VGI – which is spatial data collected by laypersons – is an important component of PBR. This reliance on non-professional data collection is what differentiates VGI from traditional forms of data. As recently as the late 20th century, access to GIS technologies was largely limited to geospatial professionals trained in data collection and analysis. With the rapid evolution of technological capabilities, the societal scope of spatial technology usage has expanded significantly. Today, many people carry advanced technology with them at all times, in the form of a smartphone, and are able to gather spatial data with much more ease than in previous generations.

VGI represents one key way in which people can now participate in data collection. VGI is differentiated from traditional geographic information in that lay citizens are the collectors of information, rather than trained individuals with academic or government institutions and private companies (Elwood 2008).

2.1.2. VGI Data Quality and Cycling

Due to VGI being sourced from non-GIS professionals who lack training in geographic and cartographic techniques, concerns have been raised regarding the accuracy and, hence, utility of VGI data.

Girra, Bédard, and Roche (2010) explain that one concern with VGI is that there is insufficient monitoring by trained GIS professionals. VGI data quality considerations are impacted by the context in which VGI is collected and utilized. For this project, VGI data quality should be analyzed within the framework of use by the bicycling community and the needs of cyclists in Philadelphia. Kessler (2011) examines cyclists' use of GPS-enabled technologies for VGI collection as a method for analyzing potential issues regarding VGI in general. In his study, Kessler contends that cyclists are inherently more likely than non-cyclists to quickly learn how to contribute to VGI efforts. Kessler (2011) underscores this point succinctly with his observation that, “[s]patial awareness is a vital part of riding a bike.” By already possessing a developed spatial awareness, cyclists have the potential to quickly join VGI efforts, even if they have little to no experience with GPS-enabled devices and GIS in general.

Some of the cycling-specific challenges Kessler identifies are not issues that this project is likely to encounter. For example, in analyzing *MapMyRide*, an app that allows users to track their cycling trip and assign it a difficulty level, Kessler points out that difficulty level is a rather subjective measure, and someone relying on VGI data that lists a ride as “easy” might become frustrated if that ride is actually quite difficult for them. While this is certainly a valid concern, it is not relevant to this project, as PBR only collects objective data, rather than subjective opinions such as difficulty level.

Other characteristics of Philadelphia's cyclists are beneficial to VGI participation. As can be seen by the activity level of the Philadelphia cycling community on Twitter, a strong culture

of information sharing and activism already exists. This enthusiasm for collaborative efforts bodes well for the potential success of a cycling app for VGI collection. The strong culture of sharing among cyclists is supported by a 2007 study (Priedhorsky, Jordan, and Terveen) that surveyed 73 cyclists on their attitude towards information sharing. The results found that 83% of participants reported a willingness to contribute to correcting map errors and to receiving route planning from other cyclists. On the whole, the authors found that study participants were generally eager to share information on cycling conditions.

2.1.2.1. Socioeconomic and Demographic Considerations

When discussing VGI data quality, it is also important to be aware of certain socioeconomic and demographic considerations that can impact the data. For instance, it is important to be aware of the possibility that certain demographic groups might report incidents at a greater rate than other groups, resulting in skewed data. For example, are cyclists in higher income neighborhoods more likely to own smartphones than cyclists in lower income neighborhoods? If so, it is likely that a disproportionate amount of data would be collected from higher income neighborhoods.

Based on 2013 surveys by the Pew Research Center's Internet & American Life Project, among individuals between the ages of 30 and 49, 47% of people earning less than \$30,000 a year own a smartphone, compared with 68% of people earning between \$30,000 and \$74,999, and 87% of people earning more than \$75,000. It is important to be aware of these differences, as the data collected by this project's app could be skewed by a disproportionate number of data points in higher income areas.

Differences in smartphone ownership are not just restricted to income levels; there are also differences with regards to age group. According to the same study mentioned above,

younger age groups are more likely to own a smartphone than older age groups. As a result, PBR is likely to be used disproportionately by younger people, which could skew the data. For example, neighborhoods in which a large number of young people live – such as University City – might receive more reports of blocked bike lanes than other neighborhoods. As such, the resulting summary maps could lead the viewer to assume that blocked bike lanes are more common in University City than anywhere else in the city, even though that might not be the case.

These disparities in smartphone ownership by income group and age group are not unrelated trends. Interestingly, the range of smartphone ownership by income group is higher in older age groups than in younger groups, as can be seen in Figure 6.

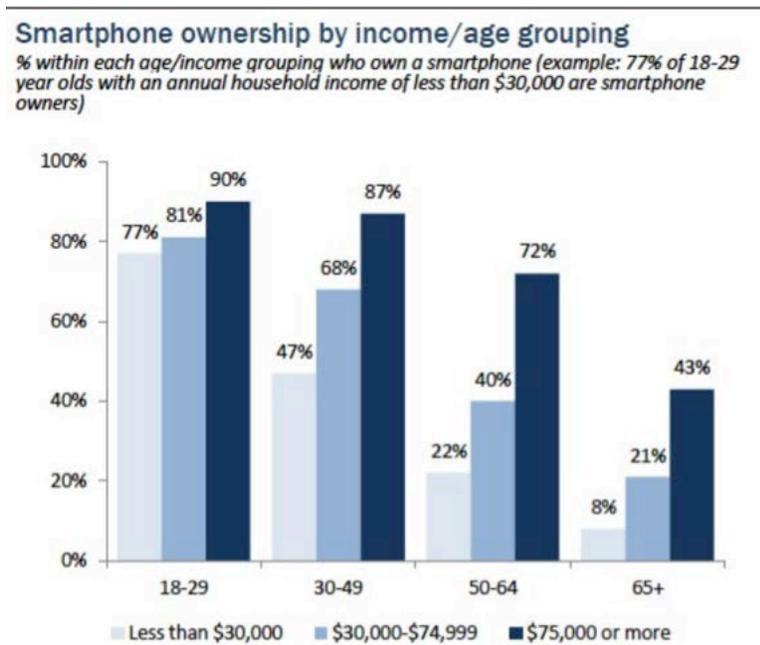


Figure 6. Smartphone Ownership by Income and Age (Pew Research Center's Internet & American LifeProject, 2013)

2.2. Existing Mobile Cycling Apps

The design and development of this project is partly informed by existing apps, particularly apps that involve cycling or VGI collection. Several cycling apps that incorporate VGI, including *BCApp*, *Cyclopath*, *BikeMaps*, and *MyBikeLane*, are discussed in the following sections.

2.2.1. *BCApp*

BCApp, by former USC GIST student Patricia Jula, allows users to count bicyclists in Los Angeles in an effort to improve upon the traditional pen and paper count method. The mobile app also includes data on collisions between bicycles and motor vehicles. Jula's thesis was useful in the planning process for this project, as the way in which the technology and programming requirements were discussed was quite helpful; Jula sequentially lists the different application requirements along with their corresponding technological descriptions.

BCApp focuses on VGI collection by people monitoring cyclist frequency, not VGI submitted by cyclists themselves. Other apps allow individual cyclists to contribute information about specific trips.

2.2.2. *Cyclopath*

Cyclopath is an app that collects VGI data from the cycling community by providing "...an editable map where anyone can share notes about roads and trails, enter tags about special locations, and fix map problems (GroupLens Research 2013)." Due to its high level of editability, the authors *Cyclopath* refer to it as a "geowiki." This thesis does not refer to PBR as a geowiki, but based on the definition provided by *Cyclopath*, it could be classified as such.

2.2.3. *BikeMaps*

Another similar app, *BikeMaps*, allows users to report collisions, near misses, cycling hazards, and bike thefts. Additionally, users are able to toggle overlays for additional information, including available ridership data and infrastructure. The overall design and functionality of *BikeMaps* provided a useful example for this project. The app includes an intuitive user interface and is easy to navigate. The navigation bar includes a link to the “Visualization” page, which provides the user with filterable summary charts and graphs. While this thesis has considerable overlap with *BikeMaps*, there are certain functions it provides that are not offered by *BikeMaps*. First, *BikeMaps* is a global service, so it does not have a focus on the popular Philadelphia-based #UnblockBikeLanes Twitter tag that helped inspire creation of this project. PBR is focused on user reports of blocked bike lanes and includes tweets of the #UnblockBikeLanes Twitter feed, neither of which are offered by *BikeMap*. Second, based on reports of cyclist hazards and collisions that can be found on Twitter, there is a substantial amount of missing data on *BikeMaps*, as many of the incidents reported on Twitter are not represented on the map. At the time this was written in February 2017, *BikeMaps* only had fifteen incidents recorded in Philadelphia, all of which occurred in 2014 or 2015, which reveals that the service is not widely used by Philadelphians. By reaching out to the Philadelphia cycling community and regularly monitoring the community's more prominent Twitter accounts, this project aims to maintain a more complete Philadelphia-focused dataset than is currently offered by *BikeMaps*. A screenshot of *BikeMaps* zoomed in to Philadelphia can be seen in Figure 7, which shows the limited number of incidents displayed by the app.

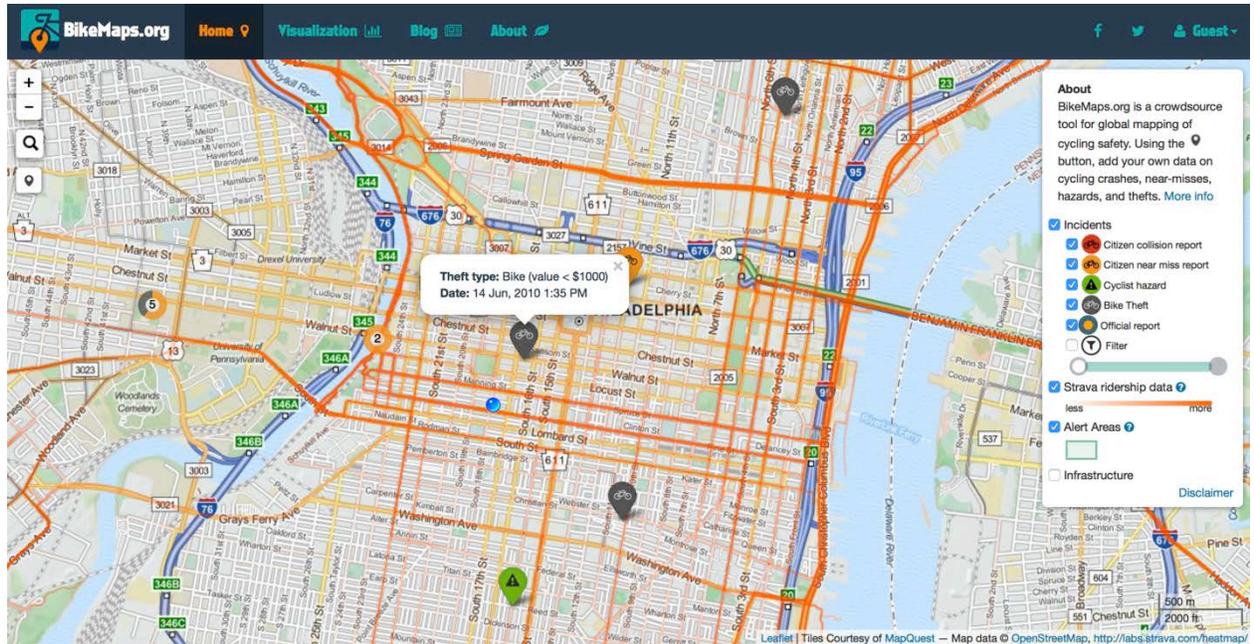


Figure 7. *BikeMaps* Web Version Homepage

It should also be noted that *BikeMaps* does not have a strong focus on blocked bike lanes, and the method for reporting a blocked lane is not readily apparent to the user. To report a blocked bike lane, the user must first select the Hazard incident type, and then find “Vehicle use of bike lane” in the drop down menu. While the breadth of reportable incident types is impressive, the specific needs of the Philadelphia cycling community would be better served by a stronger focus on blocked bike lanes, so that people can report incidents more quickly and easily, particularly while biking. As can be seen in Figure 8, the four possible types of incidents users can report with *BikeMaps* are Collision, Near miss, Hazard, and Theft.

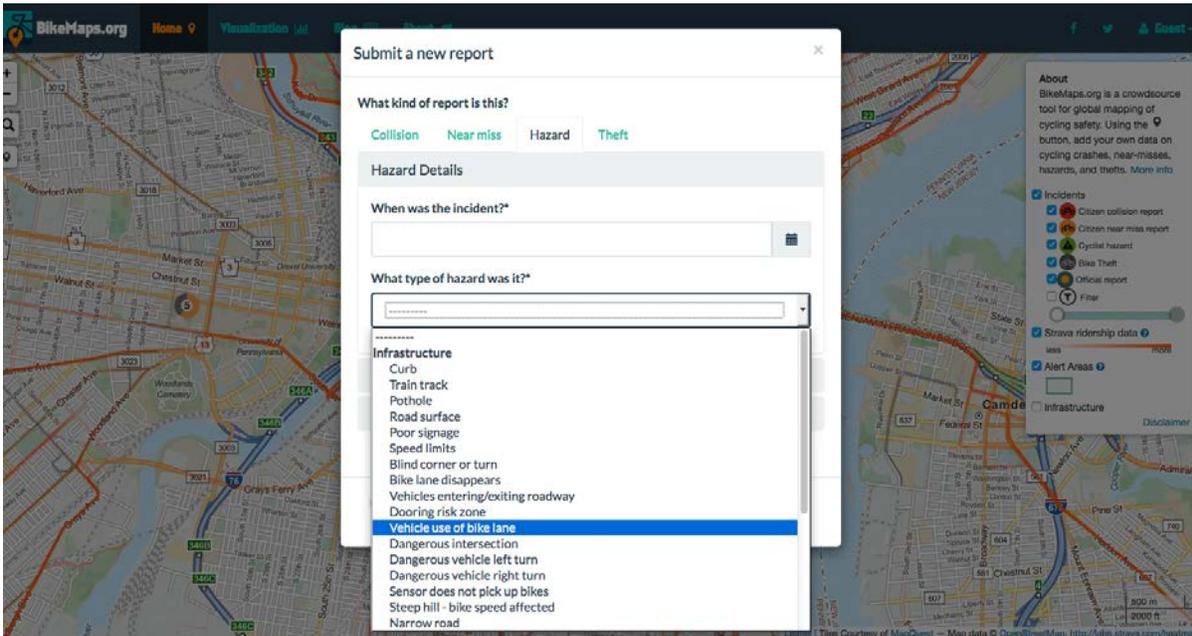


Figure 8. *BikeMaps* VGI Collection

2.2.4. *MyBikeLane*

MyBikeLane (Turnbull 2015) is an app that allows users to report blocked bike lanes in the greater Toronto area. The project appears to be successful, as it has been reported on by multiple local news outlets and continues to remain active. The success of *MyBikeLane* in Toronto bodes well for a similar project in Philadelphia.

Figure 9 shows screenshots of three different activities in the *MyBikeLane* mobile app: the Report activity, the Map activity, and the Violations activity. As can be seen in the Violations activity on the right hand side of the figure the mobile version of the app contains significantly fewer details than the web version, as the mobile app does not include readily available summary information as the web app.

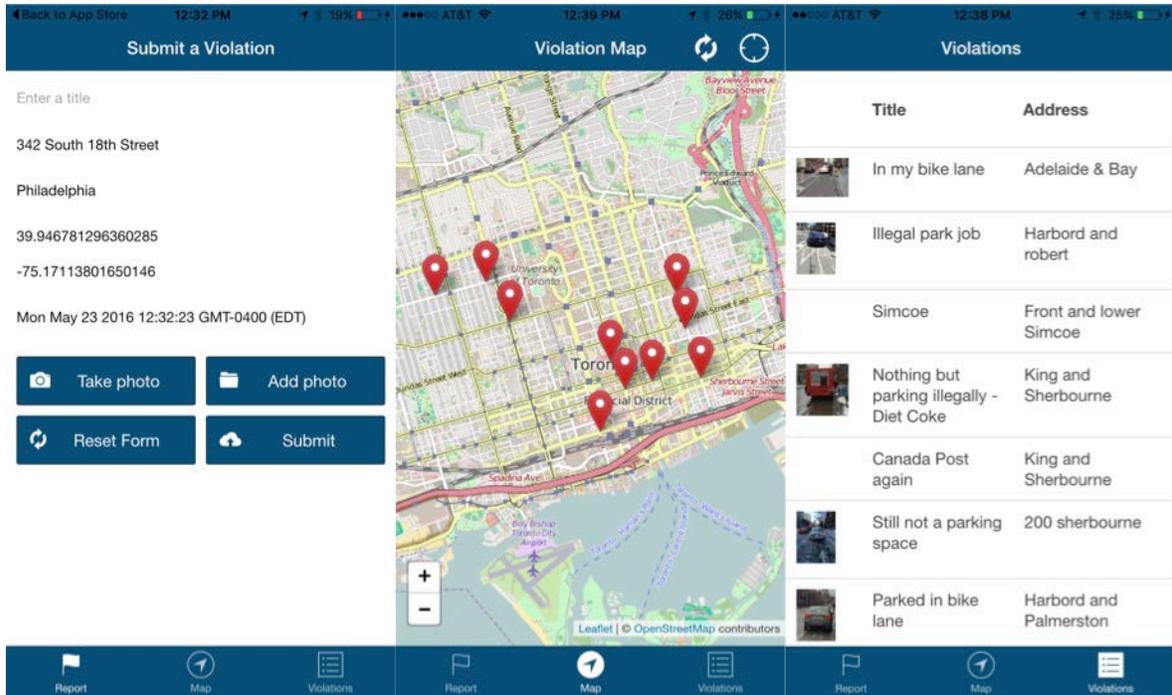


Figure 9. *MyBikeLane Mobile App*

In comparison with the mobile app, the web version of *MyBikeLane* is more extensive in its available features. The web version includes additional information, such as helpful lists of “Most Violating Organizations” and “Worst Repeat Offenders”, as shown in Figure 10, as well as summary information for a single organization, as shown in Figure 11. Another intriguing aspect of *MyBikeLane* is that it tracks vehicles that have blocked bike lanes more than once. The app’s database includes information on license plate numbers, as well as commercial vehicle numbers.

MyBikeLane Toronto Home Submit a Violation About Blog Login Sign Up

What can I add to make this site better? Send

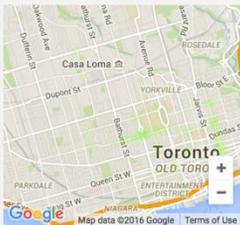
Most Recent **Worst Offenders**

 In my bike lane BBJA 536 posted 4 days ago



high traffic time (5ish in the afternoon), downtown financial district, AND no idling/stopping zone

Violations Heatmap



Most Violating Organizations

- Canada Post 24
- UPS 17
- Mark 13
- Purolator 11
- FedEx 9
- Toronto Police 6
- Coca-Cola 4
- buy amoxil 4
- Rogers 3
- Diamond Taxi 3

Worst Repeat Offenders

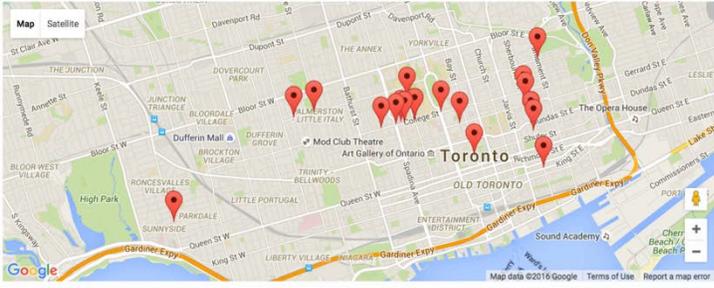
- 257 9YT 4
- 597 2NJ 4
- 739 OHP 2
- AA33220 2
- AMFN576 2
- BLSM 108 2

Figure 10. MyBikeLane Mobile App

MyBikeLane Toronto Home Submit a Violation About Blog Login Sign Up

What can I add to make this site better? Send

Canada Post first seen Thu, 16 Apr 2015 18:33:22 +0000



Organization Name Canada Post **Total Known Offences** 23

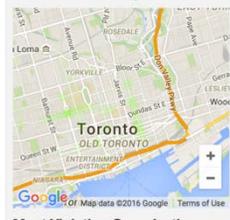
Total Known Violators 25 **Type** TODO

No description provided

Recorded Violations

Points	Title	Address	Offender	When
1	Delivering the mail	Sherbourne and college, Toronto	798 1rv	about 3 years ago
0	Just one of many	Parliament and Wellesley, Toronto	798 2RV	almost 2 years

Violations Heatmap



Most Violating Organizations

- Canada Post 24
- UPS 17
- Mark 13
- Purolator 11
- FedEx 9
- Toronto Police 6
- Coca-Cola 4
- buy amoxil 4
- Rogers 3
- Diamond Taxi 3

Worst Repeat Offenders

- 257 9YT 4
- 597 2NJ 4
- 739 OHP 2
- AA33220 2
- AMFN576 2
- BLSM 108 2

Figure 11. MyBikeLane “Repeat Offenders”

The success of *MyBikeLane* provides an example of the kind of positive impact that PBR has the potential to achieve. The *MyBikeLane* web and mobile app has received considerable media attention since its launch in 2006, including coverage by *StreetsBlog* (2006), *BlogTO* (2007), *Toronto Sun* (2014), and *The Globe and Mail* (2014). The success of *MyBikeLane* is further evidenced by the work of Mahmood (2014) who uses the app as example of an emerging mobile and Web 2.0 technology in his book about how such technologies are improving the ability of citizens to participate in governance.

2.2.5. Other Existing Cycling Apps

In addition to the apps mentioned above, there are other non-VGI-related cycling apps that were helpful in the development of PBR. One such app is *Bike2Go*, which is a mobile app for users of the Indego bike share system in Philadelphia. The app displays the location of Indego stations, as well as information of availability at each station. While the app's functionality is rather limited, it is well designed and very user-intuitive. As such, the app was useful in helping to determine the visual style of this project's app. Two sample screenshots of *Bike2Go* can be seen in Figure 12.

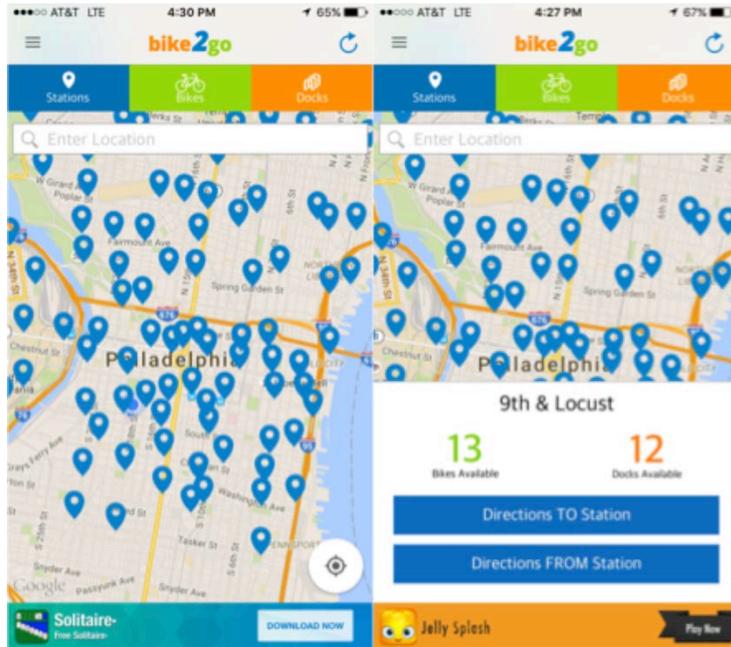


Figure 12. *Bike2Go App*

Another cycling app that was helpful to the design of PBR is *Social Cyclist* by Social Bicycles (SoBi). *Social Cyclist* includes some of the user-individualization features that this project hopes to implement in the future, including customized profiles and the ability to maintain friend lists. A sample *Social Cyclist* screenshot can be seen in Figure 13.

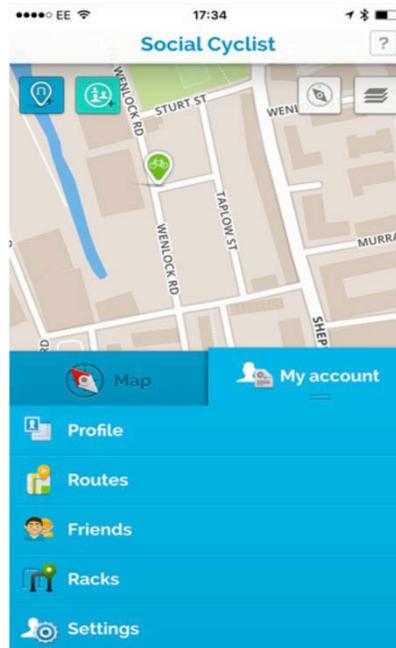


Figure 13. *Social Cyclist Mobile App*

Chapter 3 Methodology

This chapter focuses on the central objectives of the project and the planning process utilized to achieve those objectives, including consideration of intended use cases for PBR and consideration of user requirements. The chapter also explores the software and database selection process, including discussion of Backend as a Service (BaaS) providers and database hosting options. In this context, database hosting refers to which cloud-based service is utilized to store app data, while BaaS refers to which service is used to connect the app to the database. PBR can be defined as a client-server app, with a client-side – also known as a frontend – that the user sees directly on the screen of their Android device, and a server-side – also known as a backend – that refers to the data management that occurs between the app and its database, and is not directly seen by the user.

Developers can manage an app's backend by either using a BaaS provider or creating their own custom backend. Compared with using a BaaS provider, creating a custom backend often necessitates greater time investment and ongoing maintenance. Indeed, according to Manglani (2016), "A custom backend takes a lot of time to build, and afterwards requires regular maintenance – and for many small apps, this cost may not be worth the benefit." Due to these considerations, it was decided that PBR would utilize a BaaS provider for backend management. Upon researching available BaaS options, it was decided that PBR would utilize Back4App, which is a BaaS platform that utilizes a MongoDB database hosted on an Amazon Web Services (AWS) server. Back4App is designed to work closely with Parse Server, and is built specifically for use with open source Parse code. As explained by the official Back4App FAQ document (2017), the advantage of using Back4App is that by using a Parse Server backend as its core, the platform can include all the benefits of a Parse backend, while also providing additional features

developed by Back4App, including full customer support, database backup and performance maintenance services, and management of user registration with existing Facebook and Twitter accounts. The reasoning behind selection of these services for PBR is discussed in the following sections.

3.1. Project Objectives

At its core, PBR aims to provide an improved method for Philadelphia cyclists to view and report blocked bike lanes, by enhancing the current method of simply tweeting to #UnblockBikeLanes. The app provides two principal services: (1) allow users to view a spatial representation of recently reported blocked bike lanes; and (2) allow users to simultaneously submit incidents to both a central database and to Twitter with the UnblockBikeLanes hashtag. Given that blocked bike lanes are a mobility issue, the most effective solution for these objectives should be mobile-focused. PBR's mobile solution is based on a thorough planning process, involving researching of similar mobile apps, prioritizing desired functions, and determining the set of technologies to be used.

3.2. Intended User Groups

When designing a mobile app, it is important to have a clear understanding of the target audience and the ways it might engage with the application. The principal intended user of PBR is someone who cycles regularly in Philadelphia, owns a smartphone, and ideally is active in the Philadelphia cycling Twitter community. Establishing a presence on Twitter brings the dual benefits of increasing awareness of the app, as well as creating connections and dialogue. PBR is linked to a Twitter account, @BikeReportPHL, to help connect with the Philadelphia cycling community. Due to the frequency of tweets about cycling in Philadelphia, the importance of Twitter integration is perhaps even stronger for this app than most apps.

PBR thus has two distinct intended user groups: (1) people who cycle regularly and *have* tweeted to the UnblockBikeLanes hashtag; and (2) people who cycle regularly and *have not* tweeted to the UnblockBikeLanes hashtag. The reason for a focus on the first intended user group is two-fold: (1) the Philadelphia cycling community has an active Twitter presence; and (2) members of that Twitter cycling community have already demonstrated a proclivity for contributing VGI on cycling conditions. The second intended user group is equally important, because one of the key features of the app – displaying reports of blocked bike lanes – can be utilized by anyone who cycles in Philadelphia, regardless of whether or not they have tweeted to #UnblockBikeLanes. By targeting this potentially large user group, PBR might reach cyclists who are interested in contributing data about cycling conditions, but who have not opted to do so via Twitter.

PBR was inspired by the level of activity of the UnblockBikeLanes hashtag and was designed with that context in mind. In addition to reporting blocked bike lanes, many Philadelphia cyclists also use Twitter to report other cycling conditions by using #BikePHL, #BikePhilly, and #CarImpunity. The existence of four popular Twitter hashtags for raising concerns about cycling conditions in Philadelphia suggests that many people who cycle in Philadelphia are willing to use technology to share information about their trips.

The cycling community's Twitter presence is important because it provides an excellent opportunity to build connections and raise awareness of the app. Certain key cycling and alternative transportation advocacy groups, including the BCGP, Plan Philly, and Fifth Square, are active on Twitter and have a high number of followers. By building connections with these groups, it is possible that they would promote the app with their own Twitter accounts, which could increase the potential user base exponentially.

3.3. User Requirements

The core required capabilities for PBR are based on the aforementioned purpose of providing an effective tool for reporting and viewing blocked bike lanes. In order to most effectively address this problem, PBR enables the user to perform numerous actions, including submitting VGI about blocked bike lanes, viewing recently blocked bike lanes, and submitting tweets about blocked bike lanes. User requirements are discussed in the following sub-sections, and a summary overview can be seen in Table 1.

Table 1: User Requirements

REQUIREMENT	Description
3.3.1. Locate User	When user opens app, initial map zooms to user's location
3.3.2. Exchange Information About Blocked Bike Lanes	Report and view information about blocked lanes, including spatial data, textual descriptions, and photos
3.2.3. Tweet to #UnblockBikeLanes	Users have the option to Tweet their reported violation directly from the app to the #UnblockBikeLanes Twitter feed

3.3.1 *Locate User*

One of the central requirements of the app is to automatically locate the user whenever the app is opened. Due to the fact that users will typically be cycling at the time they report an incident, it is critical that the process for submitting is as efficient as possible, and opening the app to the user's current location is perhaps the most important part of maximizing efficiency.

3.3.2. *Exchange Information About Blocked Bike Lanes*

The core focus of this project is to allow users to submit VGI on cycling conditions in the city of Philadelphia. Users are able to submit information about blocked bike lanes they encounter including location, date and time information, text descriptions, and photographic evidence. Reported incidents are immediately uploaded to PBR's cloud database, so that users

can view them in real-time. This also allows users to instantly confirm that their submission was successful.

3.3.3. Tweet to #UnblockBikeLanes

Given the importance that many Philadelphia cycling advocates place on Twitter activism to raise awareness about the lack of cycling safety, it is important to involve these stakeholders in this app. The #UnblockBikeLanes campaign is a popular and unique component of these advocacy efforts. Therefore, PBR allows users to include a tweet to #UnblockBikeLanes when reporting a blocked bike lane to the app database.

3.4. Database Selection

This section provides an overview of the database selection process, including discussion of the various factors involved in determining which set of database technologies to use. This decision was made based on a few critical factors, including constant availability in the cloud, cost, and customizability.

PBR is designed to allow simultaneous use by many individuals, so a cloud database was necessary. Given the ever-increasing usage of mobile devices, there are many services available for cloud-based data management and storage. Data storage options that were considered include Microsoft's Azure DocumentDB NoSQL database service, Amazon's RDS relational database service or its DynamoDB NoSQL database service. When initially developing the PBR web app, a public Google Spreadsheet was used for simplicity, with a Java interface utilized to post the VGI to the spreadsheet. However, this method was deemed insufficient due to lack of scalability, as well as an inability to store images in the spreadsheet. Although Microsoft Azure and Amazon AWS could have met the database needs, and would have provided the needed scalability and

image storage, the needs of the app would necessitate a paid subscription to each service, which was not financially feasible.

3.4.1. Comparison of SQL and NoSQL Databases

One of the most important choices in the database selection process was whether to use a Structured Query Language (SQL) database or a Not Only SQL (NoSQL) database. While the differences between SQL and NoSQL databases are defined a number of different ways, one key distinction is that SQL databases utilize a relational data model while NoSQL databases are non-relational. As such, a SQL database can also be referred to as a relational database management system (RDBMS). As the name suggests, a RDBMS stores data in related tables that are linked together through foreign keys, which are values that define how rows in different tables are joined together. In comparison to SQL databases, the data model of NoSQL databases is less rigidly structured. Hashem and Ranc (2016) explain the flexibility of the NoSQL data model well when they state, “Non-relational data models often start from the application-specific queries, as opposed to relational data models. Non-relational data model will be then driven by application-specific access patterns.” In other words, NoSQL databases allow developers to design their data structure based on the unique needs of a given project. A high level comparison of relational databases and NoSQL databases can be seen in Figure 14.

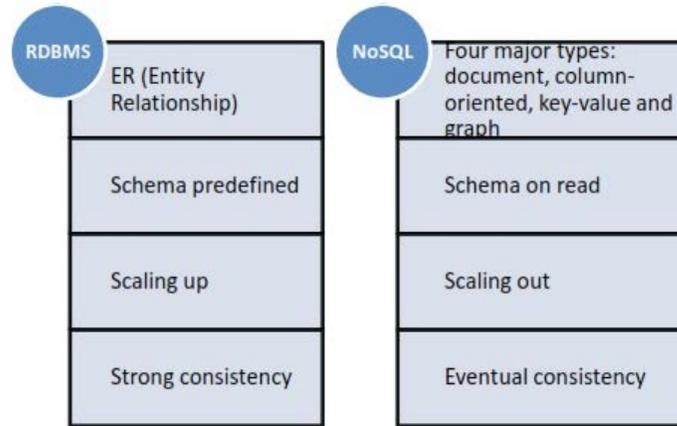


Figure 14. Comparison of RDBMS and NoSQL Databases (Hashem and Ranc 2016)

The increasing use of NoSQL databases is more pronounced in web and mobile GIS than in traditional desktop GIS. This is unsurprising, considering that one of the main benefits of a NoSQL database – the high degree of availability for multiple users to access real-time data – is often a core requirement of a web or mobile GIS tool. As explained by Altaweel (2016, 1), “web-based GIS is probably the area that is currently leading in the use of NoSQL databases within GIS, as types of real-time data are more typically found in these platforms.” Therefore, it was concluded that a NoSQL database was indeed suitable for the spatial data types used by PBR and the spatial operations that needed to be performed.

One example of a mobile GIS app that switched from a SQL database to a NoSQL database is GAIN, a personal fitness app. In September 2012, GAIN Fitness switched from PostgreSQL to Cloudant’s NoSQL database service; this decision was made due to GAIN’s desire to increase their scalability and performance during a time of quick user base growth, as well as their goal to provide improved real-time analysis and offline data sync (Cloudant 2012). GAIN’s use of a Cloudant database provides an example of how NoSQL databases provide strong support of spatial data and spatial operations.

In their research on using smartphones to create and share spatial data in a NoSQL database, Maia et al. (2016) found that NoSQL databases were indeed able to meet their data

scalability and heterogeneity needs. To compare the database types, the authors conducted simulated application tests to measure the read and write performance of a PostgreSQL 9.3 database compared to a MongoDB 3.0.3 NoSQL database. The authors also created an evaluation tool to measure the tests results. The test results found that the MongoDB NoSQL database performed significantly better for data insertion than the PostgreSQL database. Regarding read time performance, the PostgreSQL database outperformed the MongoDB NoSQL database in all instances except when there were more than 500,000 entries in the database. However, the difference in average read time – 0.2 seconds for PostgreSQL compared to 1.0 seconds for MongoDB – was significantly smaller than the difference in average insertion times. Full results of the test are shown in Figure 15.

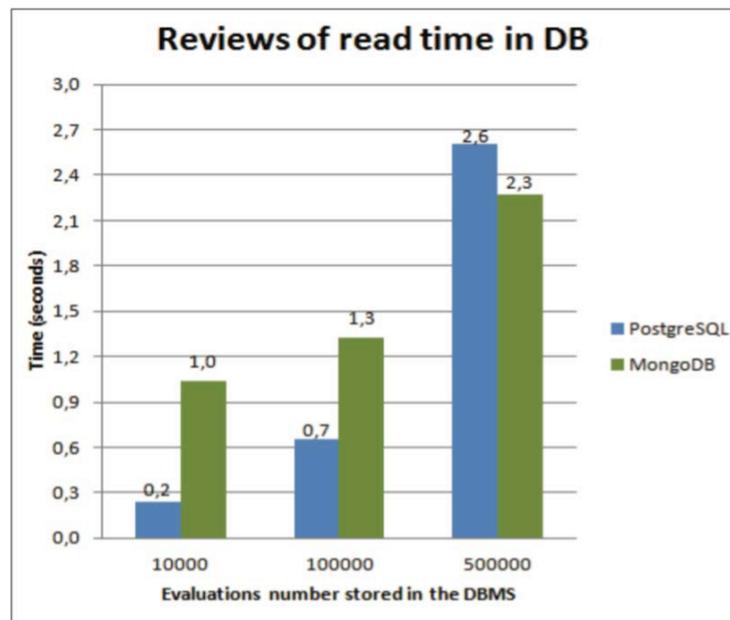


Figure 15. Read Time Comparison for SQL and NoSQL (Maia, Mendonça, Camargos, Holanda, and Maristela, 2016)

With regards to the write tests, the results further support selection of a NoSQL database for PBR, as the average insertion time for a PostgreSQL database – 21.5 seconds when the

database holds 10,000 or fewer entries – is lengthy enough that it would negatively impact the PBR user experience. It is important for PBR users to be able to verify in real time that an incident they submitted appears accurately on the map. The MongoDB NoSQL database achieved an average insertion time of only 2.8 second, which allows users to verify their reports nearly twenty seconds earlier than with a PostgreSQL database. Results for the write tests are shown in Figure 16.

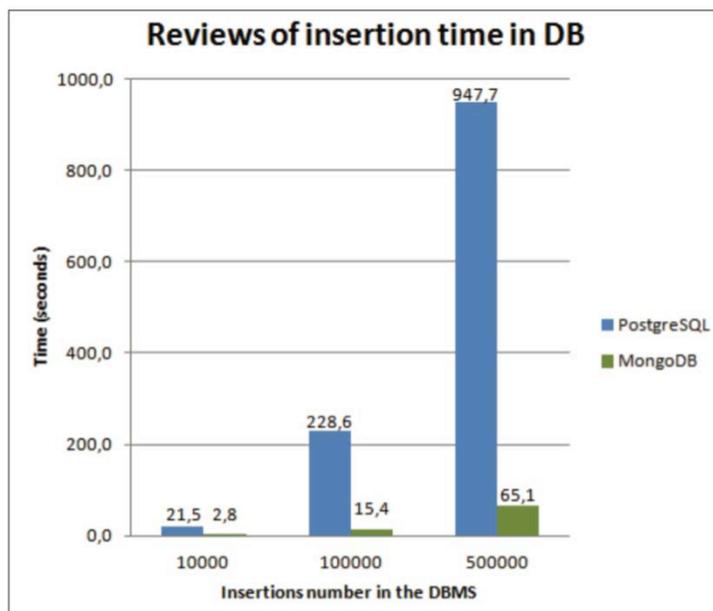


Figure 16. Write Time Comparison for SQL and NoSQL (Maia, Mendonça, Camargos, Holanda, and Maristela, 2016)

In addition to the favorable read and write times measured by Maia et al. (2016), there are other benefits of utilizing a NoSQL database for spatial data storage. In their examination of using a MongoDB NoSQL database for GIS data storage and processing, Zhang, Song, and Lui (2014) identified several advantages. Their results showed that MongoDB provided rich and efficient document querying, as well as high levels of horizontal scalability and data processing capabilities. They also found that MongoDB NoSQL databases provide strong support for spatial data types, as they allow for high flexibility with regards to data relationship management.

In another comparison study of SQL and NoSQL databases, Mao et al. (2014) analyzed the use of each database type for 3D city management systems and visualization, and found that NoSQL databases were a viable alternative to traditional SQL databases. One of their key findings that is particularly pertinent to PBR is that due to their high read/write loads and scalability, NoSQL databases are generally better suited to running on the cloud. Considering that PBR is a cloud-based app, this is an important finding. Their study also supports use of NoSQL databases for spatial data and spatial operations, as they note that many map services, including Google Maps, utilize NoSQL databases.

3.4.2. NoSQL Social Media Benefits

One key consideration in the database selection process was suitability for storing data from social media. Including this factor in the selection process provided another benefit of using a NoSQL database. The advantage of a NoSQL database in storing social media data is echoed by Altaweel (2016, 1), who states, “One reason why NoSQL data are now becoming popular are the vast quantities of unstructured or semi-structured data generated by popular social media platforms and websites.” This consideration is not especially important for the current version of PBR, as the app does not store social media data directly in the database. However, it will become more important as PBR adds new functions in the future, particularly the mapping of geotagged #UnblockBikeLanes tweets not submitted through PBR, as discussed in the Future Work section of Chapter 6.

3.4.3. NoSQL Database Selection Criteria

In addition to the question of whether to use a SQL or NoSQL database; another database decision needed to be made: which NoSQL database service to use. This decision was influenced by a number of factors, including:

- Support of Document Store database
- Integration with Parse Server
- Cost
- Reliability

First, a NoSQL database of the document store type was a criterion as this is the database type best suited for the needs of PBR. Document databases were decided upon because they allow documents to be saved in XML, JSON, or BSON formats, and also because they enable efficient and streamlined database design and management, as document databases utilize a self-defined hierarchical data structure (Hashem and Ranc 2016).

Second, a database service suitable with Parse Server was needed, as PBR utilizes Parse Server to perform many of its critical backend management functions. Parse Server is discussed further in section 3.4.4.

Third, cost was a key consideration, given that PBR is a student project. The price range of different database services is quite wide, with certain services aimed at the individual or small-scale level, and other services aimed at the enterprise level. Due to PBR's limited scope and available funding, only affordable small-scale plans were considered.

Fourth, reliability was another key factor, as PBR needs to be an efficient and reliable service in order to maximize its effectiveness for cyclists.

These factors were used to compare three services that allow integration for Parse Server, Back4App, NodeChef, and Sashido. A comparison of these three services can be seen in Figure 17.

Description	Back4App	NodeChef	Sashido
Price	Free Starts at \$4.99/month	Paid Only Starts at \$9.00 / month	Paid Only Starts at \$4.95 / month
Backup	Yes Every Hour	Info not available website	No
Redundancies	Yes	Yes (Need to pay extra)	Info not available website
Hosting	Amazon AWS	Info not available website	Cloudstrap / AWS
Vendor Lock-In	No	No	No
Support	24 / 7	8 / 5	24 / 7
Auto Scaling	Yes	No	Yes
US Legal Entity	Yes	Yes	No
FAQ	Yes	No	Yes
Docs	Yes	Yes	Yes
Blog	Yes	Yes	Yes
Partnerships with Cloud Services	Yes	No	No
Migration Engine	Yes	No	Yes
Parse Server Dashboard	Yes	Yes	Yes
Push Notifications	Yes	No	Yes
Global Config	Yes	No	Yes
Facebook & Twitter Integration	Yes	No	Yes
Background Jobs	Yes	Yes	Yes
Cloud Code Tool	Yes	Yes	Yes
Automatic Emails	Yes	No	Yes

Figure 17. Comparison of NoSQL Database Services (Batschinski 2016)

As can be seen in the comparison, Back4App is the only service that offer a free plan. Additionally, it is the only service that includes every feature listed in the comparison. A few features that are important for PBR are not supported by Back4App's competitors, as NodeChef does not include Facebook and Twitter integration, while Sashido does not offer database backup. Based on these factors, it was decided that Back4App was the ideal service for PBR.

3.4.4. Parse Server Overview

Parse Server includes a number of features that are particularly useful to PBR. By using Parse, developers are able to outsource much of the backend work that is required for a cloud-based application to function. Initially, the intention was for PBR to use Parse.com for both hosting and the backend. However, due to the planned closure of Parse hosted services on

January 28th, 2017, a different hosting service was needed, thus PBR utilizes Back4App, as discussed in the preceding section.

While there was considerable anxiety among many developers about the closure of Parse.com, the switch to Parse Server has allayed many of these concerns, in part due to the community-based open source nature of Parse Server. Indeed, in less than one year, 600,000 developers have already used Parse Server, and the Parse Server Github page already has 11,000 stars and 3,000 forks (Batschinski 2016). The high adoption rate indicated by these numbers bodes well for the future of Parse Server and provide reassurance that utilizing the service for critical function of PBR is a sound decision now and moving forward. In fact, some have argued that there are actually many benefits to the transition from Parse.com to Parse Server, as outlined in Figure 18.

Features	Parse	Parse Server
Local development & testing	No	Yes
Host in Europe, Asia, or anywhere	No	Yes
Control over database backup/restore	No	Yes
Control over database indexes	No	Yes
Query more than 1,000 objects	No	Yes
Store files elsewhere (i.e. CDN)	No	Yes
Strictly enforced time limits	Many	None
Open sourced and fully extensible	No	Yes
External contributions accepted	No	Yes

Figure 18. Hosted Parse and Parse Server Comparison (Parse 2016)

As can be seen in the above chart, the open source nature of Parse Server results in many benefits of Parse.com. With Parse Server, developers have greater control in many important ways, including extensions, database management, queries, and testing processes.

In addition to using a hosting service, Parse Server can also be used with a self-hosting instance on services such as AWS or Azure. However, dedicated instances such as these are far more expensive than shared instances on a hosting provider (Batschinski 2016). Therefore, a shared instance was the chosen option for PBR.

If PBR did not utilize Parse, this project would have taken significantly longer to complete, as Parse does much of the heavy lifting in the backend that would be quite time consuming for an individual developer. Indeed, the main goal of Parse is to empower individual developers, as the Parse website states, “The concept behind Parse has always been a simple one. Abstract away almost the entire process, allowing a solo mobile developer to build the next great mobile app (Marotto 2016).” Parse Server accomplishes this goal by using MongoDB with the Node.js JavaScript runtime environment. The prerequisites to use Parse are Node 4.3 and MongoDB 2.6.X or 3.0.X (Parse Guide 2016).

Parse is used by many mobile GIS apps. A good example is the AnyWall Android & iOS app that Parse created as a sample. AnyWall is a social app that utilizes the Google Maps API and a Parse backend, and it shares many of the functions that were included in PBR.

Due to the focus on social media integration, Parse is very helpful, as it provides tools for managing user authentication and management, as well as very helpful Twitter utilities. Using Parse, users can either register with their email address, or with their existing Twitter or Facebook accounts linked to their device.

Chapter 4 Programming And Design

This chapter focuses on the application development process, including programming languages used, app structure, app programming, and database management. The results of the process discussed in this chapter can be found in Chapter Five, which examines the degree to which the project objectives were met.

4.1. Programming Languages and Technologies

PBR was developed for Android devices, and was built using Android Studio. Android is developed using Java, thus PBR was built using Java. In addition to utilizing Java for coding of the core functions, Android development also uses Extensible Markup Language (XML) for layout design.

4.1.1. Development Environment

PBR was programmed utilizing Android Studio, which is an Integrated Development Environment (IDE) created by Google. Android Studio is the official IDE for Android, and includes a number of useful development tools, such as an intelligent code editor based on IntelliJ, a robust Android emulator, and an “Instant Run” feature, which allows developers to view coding changes without restarting the app. The Eclipse IDE was also considered. Eclipse uses a plugin, Android Developer Tools (ADT), to provide Android support, but this tool is no longer recommended. In fact, according to official Google documentation, “The Eclipse ADT plugin is no longer supported per our announcement. Android Studio is now the official IDE for Android, so you should migrate your projects to Android Studio as soon as possible (<https://developer.android.com/studio/tools/sdk/eclipse-adt.html>).” Additionally, due to the fact Google recommends using Android Studio for app development, the majority of reference

materials and tutorials are written based on the assumption that developers use Android Studio. Accordingly, Android Studio was chosen over Eclipse as the IDE. The minimum technical requirement for the app is a device with Android 4.0 (API Level 14) or higher. The development process uses Android Virtual Devices (AVD) in Android Studio, principally with a Nexus 6P AVD using API Level 23. In addition, two real devices were used for development, a Samsung Galaxy S4 smartphone, and a Samsung Galaxy Tab 3 tablet.

4.1.2. Google Maps Android API

PBR utilizes Google Maps as its mapping service, largely due to the general public's familiarity with Google Maps, as well as the generous free usage limits provided by Google. While other mapping options could have been used, the app aims to maximize ease of use by non-GIS professionals, who are not as likely to have experience with lesser-known mapping services.

Given the range of financial costs associated with available mobile GIS services, it is important to have an accurate understanding of this app's expected usage rate of its chosen service, Google Maps. Fortunately, the Google Maps Android API allows unlimited free usage, the Places API for Android allows 1,000 free requests per day, the Street View Image API allow 25,00 map loads per day, and the Geocoding and Geolocation API allow 2,500 requests per day (Google Maps APIs Pricing and Plans). These limits are more than sufficient for the needs of PBR.

4.2. App Programming

This section discusses the programming of PBR, by explaining how specific functions of the app were realized. Topics covered include user management, determining the user location, displaying incidents from the database, reporting incidents to the database, displaying recent tweet, and submitting new tweets.

It should be noted that, in order to most accurately explain how PBR was programmed, this section utilizes certain terminology that is based on Android protocols and standards, including Activity, Fragment, View, and Interface. Quick definitions of these terms, as provided by the official Android Developer Guide (2016) are as follows:

- Activity – A single screen in an application, with supporting Java code, derived from the Activity class.
- Fragment – A piece of an application's user interface or behavior that can be placed in an Activity.
- View - An object that draws to a rectangular area on the screen and handles click, keystroke, and other interaction events. A View is a base class for most layout components of an Activity or dialog screen.
- Dialog screen - A floating window that acts as a lightweight form. A dialog can have button controls only and is intended to perform a simple action (such as button choice) and perhaps return a value.

4.2.1. App Organization

PBR is organized around the main map screen, a Java class named *MapsActivity*. Rather than using additional Activities for VGI submission and data layer management, PBR utilizes Fragments within *MapsActivity*. This serves two purposes: First, it reduces unneeded complexity

by minimizing the number of Activities. Second, and perhaps more importantly, it provides a more intuitive and efficient user experience.

The main map is created using the Google Maps Android API. Initializing a Google Map is a rather straightforward process in Android involving four key steps: (1) define the map's space; (2) obtain a Google Maps Android API key; (3) provide the app with necessary API key information; and (4) modify the default map settings, as desired.

First, the map's space can be defined either as the activity itself, or as a fragment within an activity. To define the map as the activity itself, one simply selects the Google Maps option when creating a new project in Android Studio. Initially, that was the method this project used, but as development progressed, it became apparent that the other method of utilizing a map fragment was a more complete solution, due to the greater level of customization permitted by placing the map in a fragment. The XML code used for the main map fragment is shown in Figure 19 below.

```
<!--Map fragment-->
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  xmlns:map="http://schemas.android.com/apk/res-auto"
  android:id="@+id/map"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  map:cameraTilt="30"
  map:mapType="terrain"
  map:uiCompass="true"
  map:uiRotateGestures="true"
  map:uiScrollGestures="true"
  map:uiTiltGestures="true"
  map:uiZoomControls="true"
  map:uiZoomGestures="true"
  tools:context="com.bikereport.phillybikereport.MapsActivity"
  android:name="com.google.android.gms.maps.SupportMapFragment" />
```

Figure 19. XML code to determine map layout

Second, to obtain a Google Maps Android API key, one simply follows the instructions provided in the Google Developer Console to obtain the key. It is important to keep this key secret and to remove it whenever sharing source code publicly.

Third, providing the API key to the app is straight-forward: The user simply defines the API key in the `google_maps_api` XML file in Android Studio. The key also needs to be entered in the `AndroidManifest.xml` file, but Android Studio does this automatically.

Fourth, Google's standard map settings were modified to fit the needs of the project. The default map in Android Studio centers on Sydney, Australia, and includes a sample marker identifying the city. In order to customize the map for this project, the default latitude and longitude coordinates were altered to 39.9467 degrees Latitude, -75.1681 Longitude, which corresponds to the intersection of Chestnut Street and 16th Street in Center City, Philadelphia. The zoom level was increased to show greater street level detail, and the test marker variable was moved to Philadelphia and renamed from "sydney" to "philly."

4.2.2. User Management

PBR utilizes tools provided by Parse Server to manage user registration and login. Upon opening the app, a class named `DispatchActivity` makes a call to the database to determine if the device currently has a user logged in to the app. The code used is shown in Figure 20.

```
public class DispatchActivity extends AppCompatActivity {
    public DispatchActivity() {
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (ParseUser.getCurrentUser() != null) {
            startActivity(new Intent(this, MapsActivity.class));
        }
        else {
            startActivity(new Intent(this, WelcomeActivity.class));
        }
    }
}
```

Figure 20. Code to determine whether PBR has previously been launched on device

If the device is already logged in, then the user is brought directly to the main map activity. If the device is not logged in, the user is brought to welcome activity, where they can then choose between registering a new account or logging in to an existing account. User management is facilitated by utilizing the Parse Server account management tools, with data stored in Back4App. The code used for user registration can be seen in Figure 21, while the code used for user is shown in Figure 22.

```
private void signup() {
    String username = usernameEditText.getText().toString().trim();
    String password = passwordEditText.getText().toString().trim();
    String passwordAgain = passwordAgainEditText.getText().toString().trim();

    boolean validationError = false;
    StringBuilder validationErrorMessage = new StringBuilder("Merde! Error.");
    if (username.length() == 0) {
        validationError = true;
        validationErrorMessage.append("blank username");
    }
    if (password.length() == 0) {
        if (validationError) {
            validationErrorMessage.append(", and ");
        }
        validationError = true;
        validationErrorMessage.append("blank password");
    }
    if (!password.equals(passwordAgain)) {
        if (validationError) {
            validationErrorMessage.append(", and ");
        }
        validationError = true;
        validationErrorMessage.append("passwords don't match");
    }
    validationErrorMessage.append(".");

    if (validationError) {
        Toast.makeText(SignupActivity.this, validationErrorMessage.toString(), Toast.LENGTH_LONG).show();
        return;
    }
    ParseUser user = new ParseUser();
    user.setUsername(username);
    user.setPassword(password);
}
```

Figure 21. Code used to register new user

```

private void login() {
    String username = usernameEditText.getText().toString().trim();
    String password = passwordEditText.getText().toString().trim();

    boolean validationError = false;
    StringBuilder validationErrorMessage = new StringBuilder("Merde! Error.");
    if (username.length() == 0) {
        validationError = true;
        validationErrorMessage.append("blank username");
    }
    if (password.length() == 0) {
        if (validationError) {
            validationErrorMessage.append(", and ");
        }
        validationError = true;
        validationErrorMessage.append("blank password");
    }

    validationErrorMessage.append(".");

    if (validationError) {
        Toast.makeText(LoginActivity.this, validationErrorMessage.toString(), Toast.LENGTH_LONG).show();
        return;
    }

    final ProgressDialog dialog = new ProgressDialog(LoginActivity.this);
    dialog.setMessage(getString(R.string.progress_login));
    dialog.show();
    ParseUser.logInBackground(usernameEditText.getText().toString(), passwordEditText.getText().toString(), new LogInCallback() {
        @Override
        public void done(ParseUser user, ParseException e) {
            if (e != null) {
                Toast.makeText(LoginActivity.this, e.getMessage(), Toast.LENGTH_LONG).show();
            }
            else {
                Intent intent = new Intent(LoginActivity.this, DispatchActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            }
        }
    });
}
}

```

Figure 22. Code used for user login

Parse records user information as a separate class in the database, and it handles account management, include password encryption. When an incident is submitted, it is linked to the appropriate user through use of the “user” field within the “Incidents” class, which uses a unique Pointer to associate that incident with the correct object in the “User” class, as can be seen in Figure 23.

objectId	lp	comment	address	time	lat	long	user	ACL
NXd6Sag15y	abcde	fghijk	162-198 North 34th S...	11:02	39.95861146845078	-75.1910284...	nur7bAljOv	Public Re...
7bKibLacUF	PA19287	Civic	4 Lovett Avenue	0:59	39.68101631044923	-75.7496937...	o9Ishzo4CT	Public Re...
fWw9DfQl61	PA12345	Black Civic	101-117 South Colleg...	9:24	39.68159713859476	-75.7536778...	1Gcu7HV9To	Public Re...
D766u4hJTq	987	654	6020 South 24th Street	18:46	39.95114698472636	-75.1881249...	o9Ishzo4CT	Public Re...
Syp1ldEfiE	123	456	231 South 24th Street	10:14	39.95800734821...	-75.1797632...	o9Ishzo4CT	Public Re...
VkIS80EHig	(undefined)	(undefined)	55 North 22nd Street	(und...	39.95567450052267	-75.1762019...	YwTv8zmNB	Public Re...
rDw9eHZckK	(undefined)	(undefined)	2035 Chestnut Street	(und...	39.95237655771537	-75.1748245...	YwTv8zmNB	Public Re...
qNKfEuAHjg	(undefined)	(undefined)	2049 Locust Street	(und...	39.94968940834923	-75.1758703...	YwTv8zmNB	Public Re...
C23ydIVThs	(undefined)	(undefined)	116 South 21st Street	(und...	39.95127626579...	-75.1756279...	YwTv8zmNB	Public Re...
9HahXDS4ah	(undefined)	(undefined)	1598 Charleston Road	(und...	37.42113387160...	-122.082892...	YwTv8zmNB	Public Re...

Figure 23. User management in Parse Dashboard

4.2.2. Determine User Location

Upon launch the user is brought directly to their current location. This is done by using Google's Fused Location API. In the `onConnected` method of `MapsActivity`, a call is made to the Fused Location API, as can be seen in Figure 24.

```

@Override
public void onConnected(@Nullable Bundle bundle) {
    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(1000);
    mLocationRequest.setFastestInterval(1000);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
    if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
        LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest, this);
    }
    Location l = LocationServices
        .FusedLocationApi
        .getLastLocation(mGoogleApiClient);

    if (l != null) {
        Log.i("LOG", "latitude: " + l.getLatitude());
        Log.i("LOG", "longitude: " + l.getLongitude());
        LatLng localizacao = new LatLng(l.getLatitude(), l.getLongitude());
        mMap.addMarker(new MarkerOptions().position(localizacao).title("Minha Localizacao"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(localizacao));

        CameraPosition cameraPosition = new CameraPosition.Builder().target(localizacao).zoom(16).build();
        CameraUpdate cameraUpdate = CameraUpdateFactory.newCameraPosition(cameraPosition);
        mMap.moveCamera(cameraUpdate);
    }
}

```

Figure 24. Code used to locate user

Once the user's location has been determined, the map is then centered on that location, which can be seen in the second 'if statement' in the above code snippet.

As can be seen in Figure 25's first 'if statement', the `onConnected` method checks the app permission to determine if Location Services have been permitted by the user. This method is called when the app connects to the Google API client. If the user does not enable PBR to use Location Services, then the app cannot determine the user's location. When this occurs, PBR reverts to its default map position.

```
public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;
public boolean checkLocationPermission(){
    if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {

        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            android.Manifest.permission.ACCESS_FINE_LOCATION)) {
            ActivityCompat.requestPermissions(this,
                new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION},
                MY_PERMISSIONS_REQUEST_LOCATION);
        } else {
            // No explanation needed, we can request the permission.
            ActivityCompat.requestPermissions(this,
                new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION},
                MY_PERMISSIONS_REQUEST_LOCATION);
        }
        return false;
    } else {
        return true;
    }
}
```

Figure 25. Code used to determine Location Services permissions

The `onConnected` method works well for determining the user's initial location, but it does not capture any changes to the user's location. In order to determine continually updated location information that can be shown to the user, PBR contains another location method, `onLocationChanged`, that makes a separate call to the Fused Location API. The code used for this process is displayed in Figure 26.

```

@Override
public void onLocationChanged(Location location) {
    mLastLocation = location;
    if (mCurrLocationMarker != null) {
        mCurrLocationMarker.remove();
    }

    //coordinates for current location
    LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());

    //move map camera
    mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
    mMap.animateCamera(CameraUpdateFactory.zoomTo(16));

    //stop location updates
    if (mGoogleApiClient != null) {
        LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
    }
}
}

```

Figure 26. Code used to determine new location when user moves

It should also be noted that in order for the call to the Fused Location API to work correctly, GoogleApiClient.Builder must also be called, as shown in Figure 27.

```

protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
    mGoogleApiClient.connect();
}
}

```

Figure 27. Code used call GoogleAPIClient

4.2.3. Display Incidents from Database

The main map automatically displays incidents from the database upon launch. This is accomplished by using a ParseQuery to pull the data from Back4App, then calling the placeMarker method to place a marker on the map for each incident. The code used for querying incidents from the database can be seen in Figure 28 while the code used for placing the markers is shown in Figure 29.

```

ParseQuery<ParseObject> query = new ParseQuery<ParseObject>("Incidents");
query.findInBackground(new FindCallback<ParseObject>() {
    @Override
    public void done(List<ParseObject> list, ParseException e) {
        for (ParseObject p : list) {
            String lat = p.getString("lat");
            String longitude = p.getString("long");
            String address = p.getString("address");
            String date = "11/20/2016";
            String time = p.getString("time");
            String lp = p.getString("lp");
            String comment = p.getString("comment");
            ParseFile imageFile = p.getParseFile("ImageFile");

            if (imageFile != null) {
                String imageUrl = imageFile.getUrl();//live url
                Uri imageUri = Uri.parse(imageUrl);
                placeMarker(lat, longitude, address, date, time, lp, comment, imageUri);
            }
            else {
                String imageUrl = "https://parsefiles.back4app.com/hEXwBluk0amngdoKxCAAfvhmFuZ0Rc12X0TNCU58/";
                Uri imageUri = Uri.parse(imageUrl);
                placeMarker(lat, longitude, address, date, time, lp, comment, imageUri);
            }
        }
    }
});

googleMap.setInfoWindowAdapter(new PopupAdapter(this,
    getLayoutInflater(),
    images));
}

```

Figure 28. Code used to pull incidents from the database

```

//place markers from Parse db
private void placeMarker(String lat, String longitude,
String address, String date, String time, String lp, String comment, Uri imageUri) {

    double mLatitude;
    double mLongitude;
    mLatitude = Double.parseDouble(lat);
    mLongitude = Double.parseDouble(longitude);
    LatLng latLng = new LatLng(mLatitude, mLongitude);
    Marker marker=
        mMap.addMarker(new MarkerOptions().position(latLng)
            .title("placeholder")
            .snippet("Address: " + address + "\n" + "Date: " + date + "\n"
                + "Time: " + time + "\n" + "Plate #: " + lp + "\n" + "Comment: " + comment)
            .icon(BitmapDescriptorFactory.fromResource(R.drawable.blocky)));

    if (imageUri != null) {
        images.put(marker.getId(),
            imageUri);
    }
}
}

```

Figure 29. Code used to place incident markers on map

4.2.4. Submit Incidents to Database

The process for posting to the database occurs in two parts. First, a new object is created in Back4App with location data and user attribution. The code used to accomplish this is shown in Figure 30.

```
@Override
protected void onPostExecute(String addressText) {

    mMap.addMarker(new MarkerOptions()
        .position(latLng)
        .snippet(addressText));

    TextView address = (TextView) findViewById(R.id.geocoded_address);
    address.setText(addressText);

    final String lat = Double.toString(mMarker.getPosition().latitude);
    final String longitude = Double.toString(mMarker.getPosition().longitude);

    ParseObject incident = new ParseObject("Incidents");

    incident.put("lat", lat);
    incident.put("long", longitude);
    incident.put("address", addressText);
    incident.put("user", ParseUser.getCurrentUser());
    incident.saveInBackground();
}
```

Figure 30. Code used to create new object in database

Since each new object initially only has location and use data, it then needs to be edited to reflect contextual information provided by the user, including time, comments, and license plate number. The code used to execute this can be seen in Figure 31.

```

private void submitButtonListener() {
    mSubmitButton.setOnClickListener((v) -> {

        final String mComment = mCommentText.getText().toString();
        final String mLp = mLpText.getText().toString();
        final String mTime = mTimeText.getText().toString();

        ParseQuery<ParseObject> query = ParseQuery.getQuery("Incidents");
        query.orderByDescending("updatedAt");
        query.getFirstInBackground(new GetCallback<ParseObject>() {
            public void done(ParseObject object, ParseException e) {
                if (object == null) {
                    Log.d("Incidents", "The getFirst request failed.");
                } else {
                    object.put("comment", mComment);
                    object.put("lp", mLp);
                    object.put("time", mTime);
                    object.saveInBackground();
                }
            }
        });
    });
}

```

Figure 31. Code used to update object with additional information

4.2.5. Display Recent Tweets

Viewing recent tweets related to cycling in Philadelphia is accomplished using the Fabric SDK provided by Twitter. Fabric provides an adapter, *tweetTimelineListAdapter*, which can be used to set a tweet timeline list to an Android fragment. The list is populated by the search terms defined by the developer, by setting a query for Fabric's *SearchTimeline* tool. The code PBR uses to query and set cycling tweets can be seen in Figure 32.

```

final SearchTimeline searchTimeline = new SearchTimeline.Builder()
    .query("#UnblockBikeLanes OR #bikePHL OR #bikephilly OR #carimpunity")
    .build();
final TweetTimelineListAdapter adapter = new TweetTimelineListAdapter.Builder(getActivity())
    .setTimeline(searchTimeline)
    .build();
setListAdapter(adapter);

```

Figure 32. Code used to query tweets

4.2.6. Tweet to #UnblockBikeLanes

As with displaying tweets, Twitter’s Fabric SDK is also used to send tweets. PBR utilizes Fabric’s *Tweet Composer* tool, by using the code shown in Figure 33 below.

```
TwitterAuthConfig authConfig = new TwitterAuthConfig(TWITTER_KEY, TWITTER_SECRET);
Fabric.with(this, new TwitterCore(authConfig), new TweetComposer());

TweetComposer.Builder builder = new TweetComposer.Builder(this)
    .text("#UnblockBikeLanes");
builder.show();
```

Figure 33. Code used to compose tweets

4.3. Database Structure and Design

As described in Chapter 3, PBR uses a MongoDB database hosted by Back4App with a Parse Server backend. Data storage for PBR follows the default Back4App structure of storing data in a MongoDB database that is stored on AWS servers.

Parse storage is centered on the ParseObject, which is a schemaless data type that utilizes key-values pairs, and each ParseObject has a unique class name. The key for a ParseObject must be an alphanumeric string, while the value can be any data type that can be encoded into JSON, including strings, numbers, booleans, arrays, and objects.

4.3.1. Connecting to the Database

To connect to the database, the Parse API is used to make a call to the Back4App server. The code used to accomplish this is shown in Figure 34. Please note that the content of the “applicationID” and “clientKey” parameters were removed for this thesis, due to security protocol.

```
Parse.enableLocalDatastore(this);
Parse.initialize(new Parse.Configuration.Builder(getApplicationContext())
    .applicationId("REMOVED")
    .clientKey("REMOVED")
    .server("https://parseapi.back4app.com/")
    .build());
```

Figure 34. Code used to connect to Parse Server and Back4App

In addition to the code above, there are two additional steps necessary to successfully connect to the database. First, two Java Archive (JAR) files – bolts-tasks-1.4.0.jar and parse-android.1.13.1.jar – needed to be added to the app’s library folder. Second, the necessary Parse libraries needed to be imported in the app code, as shown below in Figure 35.

```
import com.parse.Parse;
import com.parse.ParseException;
import com.parse.ParseFile;
import com.parse.ParseObject;
import com.parse.ParseQuery;
```

Figure 35. Code used to import Parse libraries

Chapter 5 Results

This chapter shows the results of this project, and describes the structure of PBR, focusing on the overall organization of app and the relationships between the different app components. This is explained through a demonstration of the app, framed in terms of how PBR meets the two principal objectives described in Chapter Three: (1) allow users to view a spatial representation of recently report blocked bike lanes; and (2) allow users to simultaneously submit incidents to both a central database and the #UnblockBikeLanes Twitter tag.

Similarly to Chapter Four, please note that certain Android-specific terms are used to best explain the functions of PBR. Please refer to Section 4.2 for definitions of these terms.

5.1. App Organization

PBR is centered on the main map activity, and the main functions of viewing and reporting blocked bike lanes can both be accomplished from that activity. A flowchart explaining the components of PBR and available user actions can be seen in Figure 36.

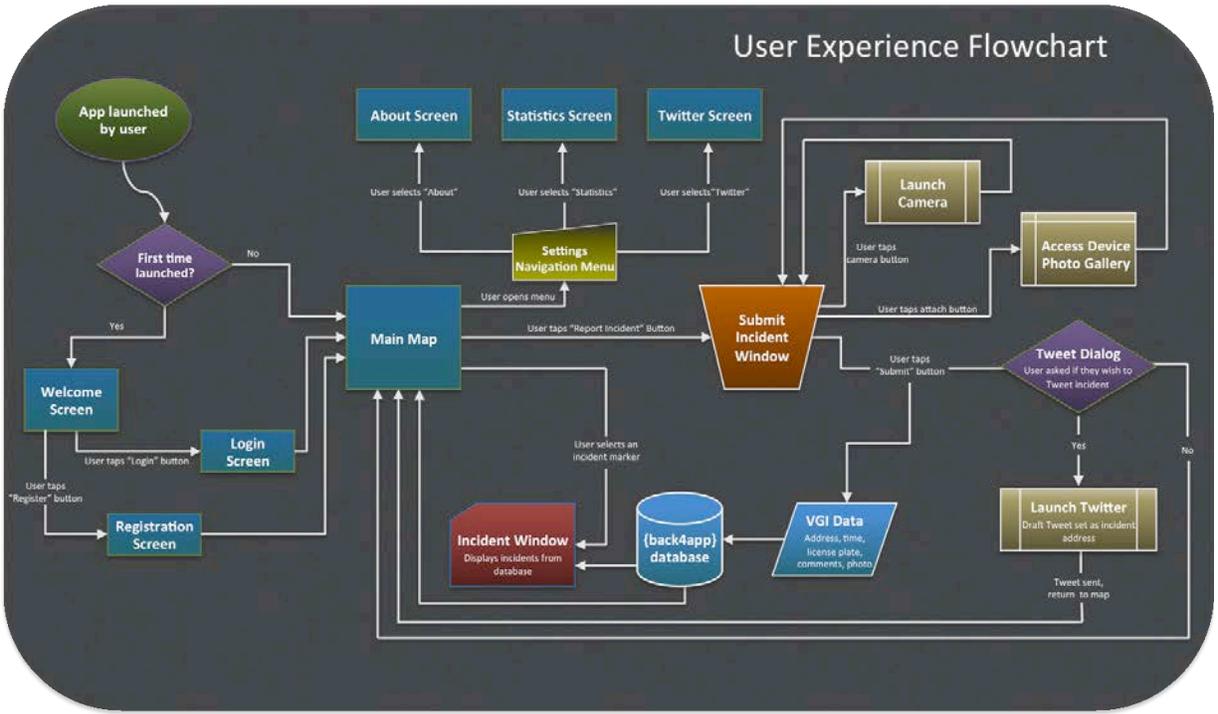


Figure 36 App Organizational Flowchart

As shown in the flowchart, upon launch PBR determines whether or not the app has previously been launched on that device. If the device has launched PBR previously then the user is brought directly to the main map activity. If PBR has not previously been launched, then the user is brought to the welcome screen shown in Figure 37. If the user has an existing an account, then the user can sign in via the login activity and if not, then they can register via the sign up activity.

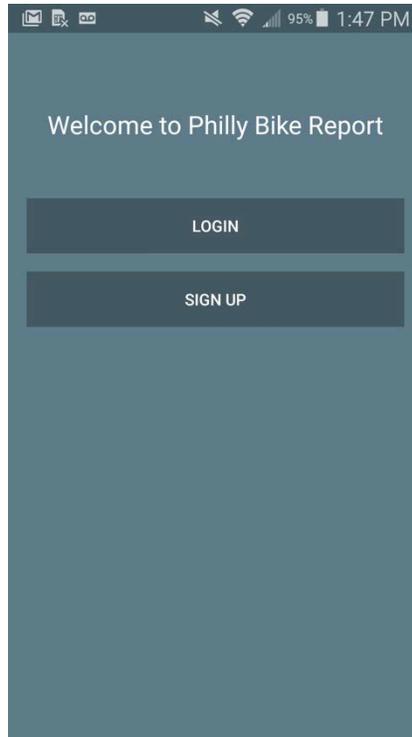


Figure 37. App Welcome Screen

After registering or logging in, the user is brought to the main map screen, with the initial map zoomed to the user's location. However, if it's the first time the app has been launched on the device, then the user receives a pop up notice asking them to enable Location Services. In order for the app to display the user's location, the user must grant Location Services permission.

5.2. Main Map

Map space is maximized by placing most of the user interface controls on the edges of the app screen, with the main report incident function accessible by pressing the "Report incident" button, zoom controls accessible by the button in the lower right corner, and all other functions accessible by utilizing the main context menu located in the upper right corner of the app bar. The blue and red circles on the map represent reported blocked bike lanes, which can be

tapped by the user to view additional information. A screenshot of the main map screen is shown in Figure 38.

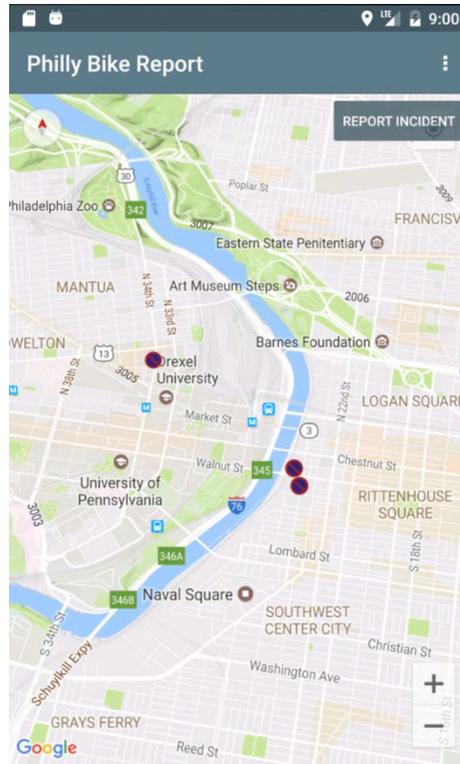


Figure 38. Main Map Screen

5.3. Incident Info Windows

When *MapsActivity* is opened, PBR automatically pulls incidents from the database and displays them as markers on the map. These markers can then be selected to view text info about the incident, as well as a photo of the incident, as shown in Figure 39. If there is no photo of the selected incident, then the info window displays a “No Image Available” placeholder as shown in Figure 40.

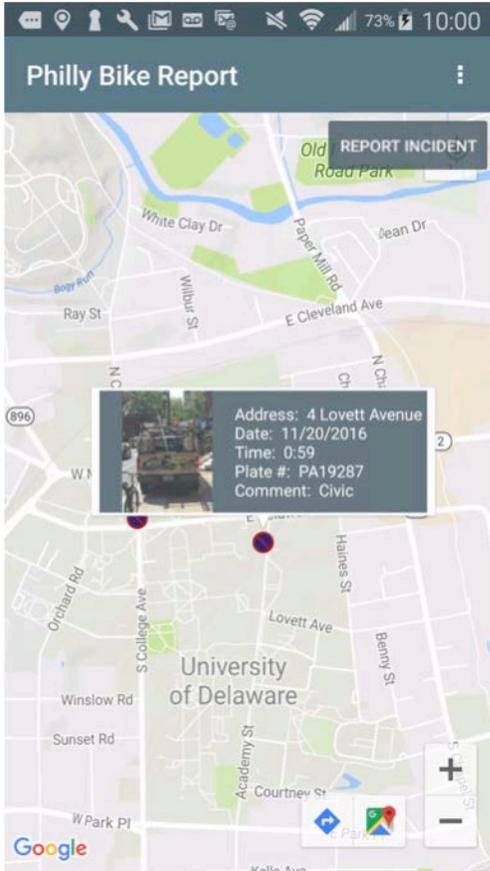


Figure 39. Incident Window A

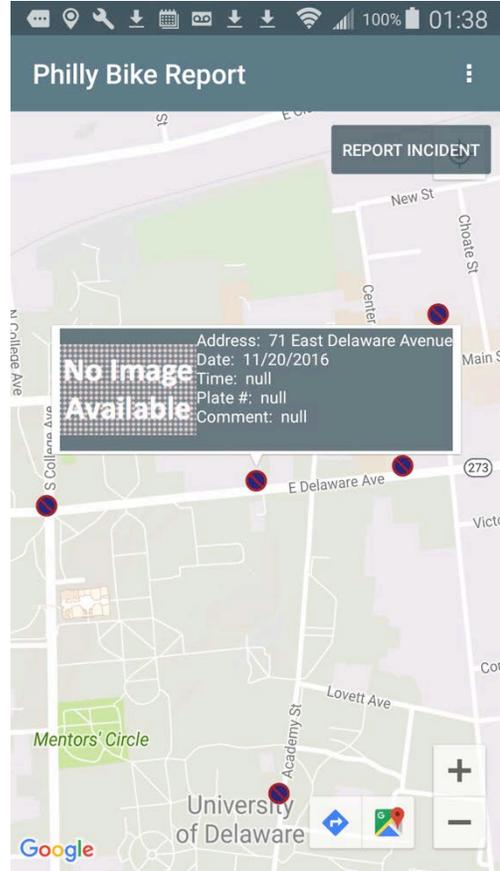


Figure 40. Incident Window B

5.4. Report Incident Fragment

When a user adds a marker, the "report incident" fragment pops up as a window within the main map. This allows the user to simultaneously enter information while still viewing the map. The address of that marker is automatically displayed, so that the user can confirm. The window also displays the current date and time. The date and time inputs are set to the current time as default. This helps support cyclists who wish to efficiently report incidents immediately, and also allows users to submit data at a later time if they prefer. Date and time scrolling pickers were also considered, but it was determined that for reporting events immediately, using scrolling pickers would be more time intensive than text fields with the current date and time set as the default entry. A screenshot of the report incident fragment is shown in Figure 41.

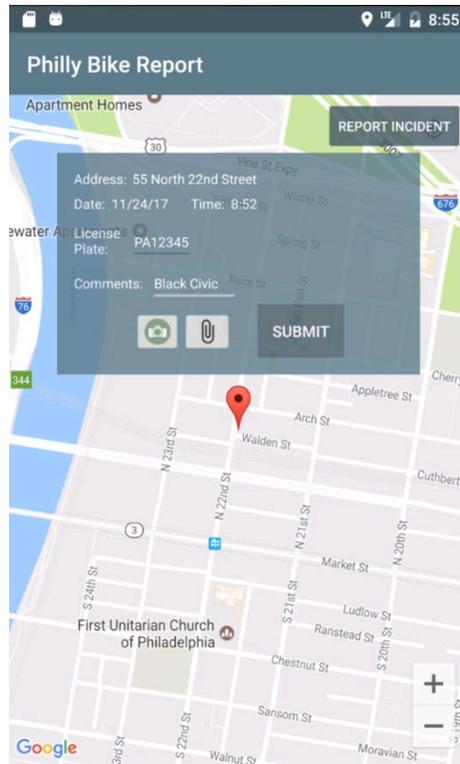


Figure 41. Report Incident Fragment

In the report incident fragment, both the address and time fields are editable, so the user can modify this information if reporting an incident that occurred at a different location or time.

In that same window, users can then add license plate information, comments, and photos. Users can choose to attach an existing photo from their storage or to take a new photo with their device's camera. The photo is uploaded to the database along with the location and other text info.

After selecting a photo, a preview is shown, and the user taps the submit button when ready. Upon clicking submit, a dialog asks the user if they would like to tweet their submission, which can be seen in Figure 42.

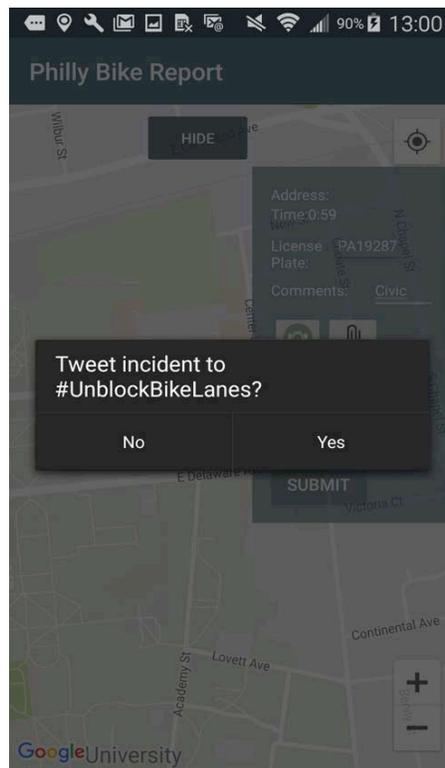


Figure 42. Twitter Dialog Prompt

If the user responds “Yes” then the Twitter app is launched with a draft tweet containing the submitted address and “#UnblockBikeLanes” set as the default text. After the tweet is sent, then the user is automatically brought back to the main PBR map. If the user responds “No” then they are brought back to the main map.

5.5. Navigation Menu and Other Activities

In addition to using the dedicated “Report Incident” button, users may also access the VGI collection activity through the app’s navigation menu. Although PBR intentionally minimized the number of Activities to maximize the user experience, there are still a few other supplementary Activities that bring the user away from *MapsActivity* to a new screen. These activities are *TwitterActivity*, *AboutActivity*, and *StatisticsActivity*. *TwitterActivity* displays recent tweets to the #UnblockBikeLanes, #BikePHL, #BikePhilly, and #CarImpunity hashtags. A sample screenshot can be seen in Figure 43.



Figure 43. Twitter Activity

AboutActivity includes information about app development, a link to USC’s GIST program, and background information about the #UnblockBikeLanes Twitter campaign and cycling in Philadelphia. *StatisticsActivity* contains summary info about data collected by the app, including where reported blocked bike lanes occurs along with any available photos of the incident. Screenshots of *AboutActivity* and *StatisticsActivity* can be seen in Figure 44 and Figure 45.

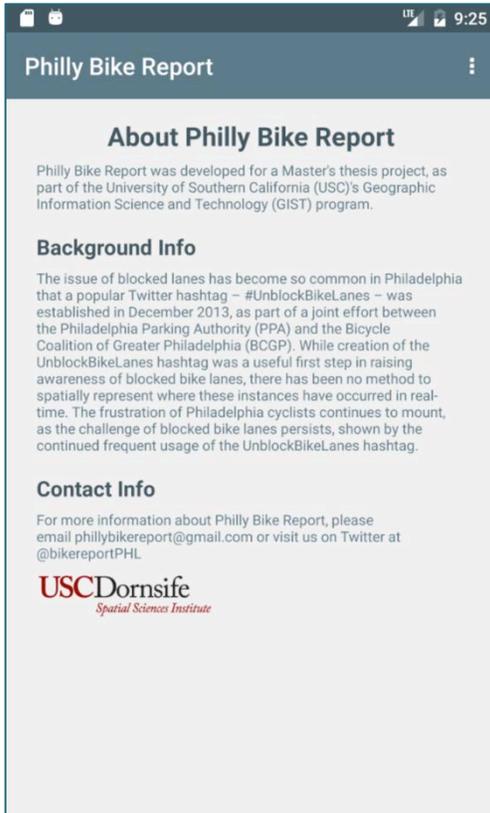


Figure 44. About Activity

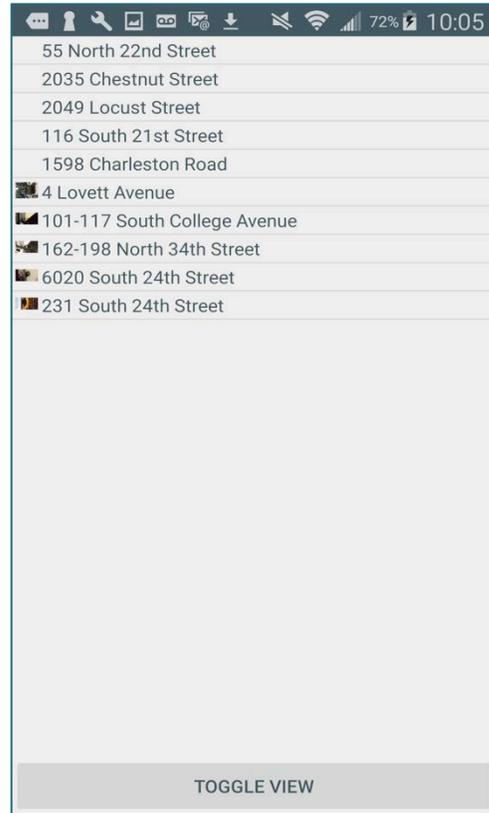


Figure 45. Statistics Activity

Chapter 6 Conclusions

This chapter includes reflection on the development process and the degree to which the goals of PBR were met. The chapter begins with discussion of this project's key finding, followed by challenges encountered, and ends with an examination of future work, including further Twitter integration, additional data layers, and support for iOS and web platforms.

6.1. Key Findings

The key findings of this project are represented by the creation of PBR as a demonstration of how a mobile app with a cloud database can be used to view and report blocked bike lanes. In assessing the results of this project, it is helpful to revisit the user requirements outlined in Chapter 3 to determine the degree to which each requirement was met. The three specified user requirements were: (1) Locate User; (2) Exchange Information About Blocked Bike Lanes; and (3) Tweet to #UnblockBikeLanes. All three of these requirements were met, as discussed below.

First, PBR successfully locates the user upon launch, and the initial map automatically zooms to the current location, thus the first requirement was fully achieved. Second, users are able to view and report blocked bike lanes with spatial data, textual descriptions and photographic evidence, so the second requirement was also achieved. The third requirement, submitting incidents to #UnblockBikeLanes directly from the app, was also successfully achieved. Upon submitting an incident to the database, users are asked if they also wish to tweet the incident. If the user responds "Yes" then a draft tweet is automatically created. However, in order to tweet a photo, the user currently has to select the photo manually. Ideally, the photo that was uploaded to the database would automatically be included in the draft tweet.

6.2. Reference Resources

There are many open source resources available online that proved valuable in solving challenges during the development process. PBR utilized several reference resources, including Google's official documentation for Android development and the Google Maps Android API, Parse's documentation for Parse Server, and guides provided by Back4App.

In addition to official documentation, community resources such as Stack Overflow and GitHub were very helpful. Stack Overflow's extensive information on a wide range of programming-related topics was very helpful, while GitHub was a useful resource for evaluating technologies that PBR could potentially utilize, as well as possible solutions to various coding problems.

6.3. Future Work

There are many additional functions that could enhance the effectiveness of PBR in the future, including additional Twitter integration, iOS and web support, and additional data layers. These topics for future work are each examined below. This section concludes with discussion of next steps for PBR, focusing on which of future work tasks will be prioritized first, and a potential timeline for additional development.

6.3.1. Further Twitter Integration

Currently, users can submit tweets and also view tweets in text format, but cannot view a spatial representation of #UnblockBikeLanes tweets that weren't submitted through PBR. PBR's integration with Twitter could be expanded by mapping geotagged #UnblockBikeLanes tweets in the app's main map. Therefore, a key future development task of PBR is displaying the location of #UnblockBikeLane tweets in the app map.

It should be noted that the majority of the tweets found in the #UnblockBikeLanes, #BikePHL, #BikePhilly, and #CarImpunity hashtags are either not related to a specific location, or do not contain georeferenced data. In order for a tweet to include location data, the Twitter user must enable the appropriate setting in their account settings.

However, even if a tweet is not georeferenced, it may still be possible to map the tweet. The ability to do so is largely dependent on the type of information given by the user. One of the more common ways of reporting incidents to the #UnblockBikeLanes hashtag is by stating the block on which it occurs, but this can lead to varying levels of accuracy. For example, in the #UnblockBikeLanes column in Figure 7, the location given in the first tweet (“2000 block Pine St”) is likely to deliver more precise data than the location given in the third tweet (“13th across from Amis”).

6.3.2. Additional Data Layers

It could be helpful for users of PBR to have access to related cycling information when using PBR. Datasets that could be helpful include bike racks, collision hotspots, and Indego bike share locations. OpenDataPhilly, which is an effort by the city government of Philadelphia to provide open access to civic data, provides the majority of these dataset. Potential datasets to be integrated into PBR in the future can be seen in Table 2.

Table 2. Potential datasets to add to PBR

DATASET	Usage	Contents	Source
Bike Network	Display cycling network	Philadelphia's bike network categorized by type	OpenDataPhilly
Bike Racks - MOTU	Display location of bike racks	Location of bike racks	OpenDataPhilly
Indego Stations	Display bike share stations	Indego Bike share stations	OpenDataPhilly
Indego Usage	Display bike share usage	Indego bike trips taken in 2015	Indego
Parking Areas	Display motor vehicle parking	Polygon data on parking lots and garages	OpenDataPhilly
Traffic Count	Display motor vehicle traffic rate	Traffic volume data on roadways in Philadelphia	Code for Philly

6.3.3. iOS and Web Support

Another area of potential future work is platform expansion beyond Android to incorporate iOS and web support. However, considering that PBR is most likely to be used by cyclists while travelling through the city, it is also important to start work on an iOS version soon after the web version is finalized, as this would allow for a greater number of cyclists to benefit from PBR while cycling.

However, it was determined that PBR would ideally be suited for mobile use, as the majority of anticipated users would submit data while cycling. The web version was much more rudimentary than the Android app discussed in this thesis, as it utilized a Google Spreadsheet for data storage, and did not include any of the Twitter functions. As discussed in the database selection section in Chapter Three, a Google Spreadsheet is inadequate for a mobile app that aims to enable simultaneous contributions by multiple users, and this is true for a web app as well.

6.3.4. Data Sharing

It is important to identify how data collected by PBR might be shared with interested members of the public. Potential interested parties include other app developers who wish to learn about PBR's methodology, as well as non-developer members of the public who might be interested in utilizing the data to assist with certain decision making processes. For example, Philadelphia cycling advocates who attend community meetings about proposed protected bike lanes might wish to use PBR's data to help strengthen their argument that improved cycling infrastructure is needed. Protected bike lane advocates could use data from PBR as evidence that unprotected bike lanes are routinely blocked.

In order to effectively share information with the public, PBR needs an easily accessible location from which users can download data collected by the app. Therefore, a key future work objective is to establish a web location that automatically syncs with the database every 24 hours. The Parse backend utilized by Back4App can be used to accomplish this goal, as the service includes a Cloud Code function that developers can use to integrate custom code. By using this Cloud Code function, PBR can implement a custom script that will automatically download data each day and then post that data to a specified location. Given the planned integration with the web version of PBR, a link within the PBR website is the ideal location to which this data should be posted.

6.3.5. Next Steps

In order to best execute the planned future work objectives, it is important to determine a timeline for which tasks should be prioritized first. Currently, the planned first next step is to complete the web version of PBR and integrate it with the Android version. There are two key reasons for this: First, the web version is already partially completed, and finishing it should not

be that time-intensive. Second, the data-sharing objective of this project necessitates a finished web version, as this is where the data will be made available.

After the web app is complete, the next planned step is to enable data sharing with the public, as discussed in the previous section. Similarly to completing the web app, enabling data sharing should be less time-intensive than the other future work objectives.

Once these objectives have been achieved, the next step is to focus on creation of the iOS version. The anticipated timeline for this task is unknown, as the author of PBR has other time commitments and limited knowledge of iOS development. One option to speed up the development process is to make the code for PBR available to members of the public by sharing it on sites such as GitHub, so that any interested iOS developer could use that code to create an iOS version.

The remaining planned development goals – further Twitter integration and addition of other cycling data layers – will be worked on after the other goals have been achieved. These goals are still important for maximizing the effectiveness of PBR, but are not as highly prioritized, because they do not expand the potential user base as much as the other goals.

References

Altaweel, Mark. "GIS and NoSQL Databases." *GIS Lounge*. May 16, 2016.

<https://www.gislounge.com/gis-nosql-databases/>.

Andersen, Michael. "14 Ways to Make Bike Lanes Better (The Infographic)." *People for Bikes*.

May 15, 2014. <http://www.peopleforbikes.org/blog/entry/14-ways-to-make-bike-lanes-better-the-infographic>.

Android Developer Guide. *Google*. <https://developer.android.com/index.html/>.

Back4App. *Frequently Asked Questions*. 2017. <http://docs.back4app.com/docs/faq-parse-back4app/>

Batista, Ramon. "Parse Server Review Migration Down to the Penny and Performance Comparison." *Medium*. October 31, 2016. <https://medium.com/@ramonvitor/parse-server-review-migration-down-to-the-penny-and-performance-comparison-dac568a6d84>.

Batschinski, George. October 30, 2016. "Parse Server Dossier – All you need to know about Parse shutdown." *Back4App*. <http://blog.back4app.com/2016/10/30/parse-server-dossier/>.

Beck, Tom. "10 Philly Streets Crying Out for Protected Bike Lanes." *Philly Mag*. June 23, 2015.

Bicycle Coalition of Greater Philadelphia (BCGP). "Better Mobility." 2016.

http://bicyclecoalition.org/wp-content/uploads/2016/01/BetterMobility03.16.15.Final_.pdf
(accessed September 4, 2016).

Bicycle Coalition of Greater Philadelphia (BCGP). "Vision Zero Philadelphia." 2015.

http://bicyclecoalition.org/wp-content/uploads/2014/01/VisionZero_Report_7.2_Web.pdf
(accessed September 4, 2016).

Bhuvan, Nikhila T and M Sudheep Elayidom. 2015. "A Technical Insight on the New

Generation Databases: NoSQL.” *International Journal of Computer Applications*
121 (7): 24-26.

Bike2Go (version 1.02). Android, iOS. Lightning Development Company. 2015.

Chen, Wei-Chen; Wu, Chao-Lin; Chen, Ya-Hung; Fu, Li-Chen. “An Efficient Data Storage Method of NoSQL Database for HEM Mobile Applications in IoT.” Paper presented at the 2014 IEEE International Conference on Internet of Things Green Computing and Communications, Taipei, Taiwan, September 1-3, 2015.

Center City District and Central Philadelphia Development Corporation. “Center City Report: Bicycle Commuting.” October 2016. http://centercityphila.org/docs/CCR16_bicycles.pdf

City of Philadelphia, Mayor’s Office of Transportation and Utilities (MOTU). 2013.

“Philadelphia Complete Streets Design Handbook.”

http://www.philadelphiastreet.com/images/uploads/resource_library/cs-handbook.pdf.

Clark, Andrew. “Steer clear of the bike lane, it's an endless cycle of frustration.” *The Globe and Mail*. May 30, 2014. <http://www.theglobeandmail.com/globe-drive/culture/commentary/steer-clear-of-the-bike-lane-its-an-endless-cycle-of-frustration/article18883518/>.

Cloudant. “Cloudant Provides Data Layer as a Service for Heroku-based GAIN Fitness Mobile Application.” September 17, 2012. <http://cloudant.com/press-releases/cloudant-provides-data-layer-as-a-service-for-heroku-based-gain-fitness-mobile-application/>.

CyclePhilly (version 1). Android, iOS, Web. Code for Philly and the Delaware Valley Regional Planning Commission, 2015.

Cyclopath (version 2.0.11). Android, Web. GroupLens Research at the University of Minnesota,

- 2013.
- Dunn, C. E. 2007. "Participatory GIS - a people's GIS?" *Progress in Human Geography* 616-637.
- Elwood, Sarah. 2008. "Volunteered geographic information: future research directions motivated by critical, participatory, and feminist GIS." *GeoJournal* 72: 173-183.
- Fishman et al. 2015. "Dutch Cycling: Quantifying the Health and Related Economic Benefits." *American Journal of Public Health* 105 (8): e13-e15.
- Fotache and Cogean. 2013. "NoSQL and SQL Databases for Mobile Applications. Case Study: MongoDB versus PostgreSQL." *Informatica Economică* 17 (2): 41-58.
- Goodchild, Michael. 2007. "Citizens as sensors: The world of volunteered geography." *GeoJournal* 69: 211-221.
- Girra, Joel; Bédard, Yvan; Roche, Stéphane. 2010. "Spatial data uncertainty in the VGI world: Going from consumer to producer." *Geomatica* 64 (1): 61-71.
- Hashem, Hadi and Daniel Ranc. "Evaluating NoSQL Document Oriented Data Model." Paper presented at the Future Internet of Things and Cloud Workshops (FiCloudW), IEEE International Conference on. 22-24 August, 2016.
- Isaac, Mike; Hardy, Quentin. Facebook to Shut Down Parse, Its Platform for Mobile Developers. New York Times. January 28, 2016. <http://bits.blogs.nytimes.com/2016/01/28/facebook-to-shut-down-parse-its-platform-for-mobile-developers/>.
- Jula, Patricia. 2015. "Generating bicyclist counts using volunteered and professional geographic information through a mobile application." Master's thesis, University of Southern California.
- Kessler, Fritz. 2011. "Volunteered geographic information: a bicycling enthusiast perspective."

- Cartography and Geographic Information Science* 38 (3): 258-268.
- Lobasso, Randy. "Support Safe Bike Lanes on December 13." *Bicycle Coalition of Greater Philadelphia*. December 8, 2016
- Macmillan, Alexandra et al. 2014. "The Societal Costs and Benefits of Commuter Bicycling: Simulating the Effects of Specific Policies Using System Dynamics Modeling." *Environmental Health Perspectives* 122 (4): 335-344.
- Madison, Michael; Barnhill, Mark; Napier, Cassie; Godin, Joy. 2015. "NoSQL Database Technologies." *Journal of International Technology and Information Management* 24 (1): 1-14.
- Mahmood, Zaigram. *Emerging Mobile and Web 2.0 Technologies for Connected E-Government*. May 21, 2014.
- Maia, Daniel Cosme Mendonça; Camargos, Breno; Holanda, Maristela. "Voluntary Geographic Information Systems with Document-based NoSQL Databases. Paper presented at the 2016 11th Iberian Conference on Information Systems and Technologies.
- Manglani, Kishin. "Top 5 Parse Alternatives." *RayWenderlich.com*. February 11, 2016.
<https://www.raywenderlich.com/126098/top-5-parse-alternatives>.
- Mao, B; Harrie, L; Cao, J; Z Wu; Shen, J. 2014. "NoSQL Based 3D City Model Management System." *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* XL.4: 169-173.
- Marotto, Fosco. "What is Parse Server?" *Parse*. March 1, 2016.
<http://blog.parse.com/announcements/what-is-parse-server/>.
- McKenzie, Brian. 2014. "Modes Less Traveled – Bicycling and Walking to Work in the United States: 2008-2012." *U.S. Census Bureau: American Community Survey Reports*.

MyBikeLane (version 0.2.9). Android, iOS, Web. Justin Bull, 2015. www.MyBikeLane.to.

Naparstek, Aaron. "Great New Website: Get Outta MyBikeLane." *StreetsBlog*. October 9, 2006. <http://nyc.streetsblog.org/2006/10/09/great-new-website-get-out-of-my-bike-lane/>

National Association of City Transportation Officials (NACTO). "High-Quality Bike Facilities Increase Ridership and Make Biking Safer." July 20, 2016. <http://nacto.org/2016/07/20/high-quality-bike-facilities-increase-ridership-make-biking-safer/>.

Okwudire, Shidi. "How I Chose My Replacement for Parse.com." *CodeNameOne*. August 9, 2016. <https://www.codenameone.com/blog/how-i-chose-my-replacement-for-parse-com-part-2.html>.

Parse Server Guide. <https://parseplatform.github.io/docs/parse-server/guide/> (last accessed November 20, 2016).

Philadelphia Parking Authority (PPA). 2015. "#UnblockBikeLanes: Continuing the Conversation." <http://www.philapark.org/2015/05/unblockbikelanes-continuing-the-conversation/> (accessed 15 February 2016).

Philadelphia Parking Authority (PPA). 2013. "#UnblockBikeLanes: New Hashtag Helps Identify Problem Spots for Cyclists." <http://www.philapark.org/2013/12/7036> (accessed 15 February 2016).

Priedhorsky, Reid; Jordan, Benjamin; Terveen, Loren. 2007. "How a Personalized Geowiki Can Help Bicyclists Share Information More Effectively." *GroupLens Research and University of Minnesota*.

Romero, Melissa. "New interactive visualizes where Philly bikes by Indego." *Curbed Philadelphia*. November 18, 2016. <http://philly.curbed.com/2016/11/18/13664054/indego-bikeshare-rides-visual-data>.

Shore, Tim. "Get Out of My Bike Lane." *BlogTO*. August 2, 2007.

http://www.blogto.com/environment/2007/08/get_out_of_my_bike_lane/

Smith, Aaron. "Smartphone Ownership 2013." *PewResearchCenter*. June 5, 2013.

<http://www.pewinternet.org/2013/06/05/smartphone-ownership-2013/> (Accessed May 25, 2016).

Spencer, George; Chang, David. "Advocates Call for Speed Cameras as Bike Deaths Rise in Philly." *NBC10 News Philadelphia*. May 24, 2016.

<http://www.nbcphiladelphia.com/investigations/Bike-Deaths-Philadelphia-Speed-Cameras-Advocates--380705421.html/> (accessed May 25, 2016).

Strobel, Mike. "Bike lane zealots a creeping menace." *Toronto Sun*. June 3, 2014.

<http://www.torontosun.com/2014/06/03/bike-lane-zealots-a-creeping-menace.>

Sui, Daniel; Elwood, Sarah; Goodchild, Michael. 2013. "Crowdsourcing Geographic Knowledge: Volunteered Geographic Information (VGI) in Theory and Practice." New York: Springer Dordrecht Heidelberg.

Tampubolon, Winhard. "Hybrid Concept of NoSQL and Relational Database for GIS Operation." Paper presented at the 19th AGILE International Conference on Geographic Information Science, Helsinki, Finland, June 14-17, 2016.

Tannenwald, Jonathan. "Philly expands bike-share program after a booming first year." *Philly.com*. April 21, 2016. http://www.philly.com/philly/news/20160422_Philly_expands_bikeshare_program_after_a_booming_first_year.html.

U.S. Census Bureau. "Means of Transportation to Work." 2012 American Community Survey 1-Year Estimates.

U.S. Department of Transportation (DOT) Federal Highway Administration (FHWA). 2015.

“Separated Bike Lane Planning and Design Guide.”

https://www.fhwa.dot.gov/environment/bicycle_pedestrian/publications/separated_biklane_pdg/page01.cfm.

Wang, Allen. Unified Data Modeling for Relational and NoSQL Databases. *InfoQueue*. February 28, 2016. <http://www.infoq.com/articles/unified-data-modeling-for-relational-and-nosql-databases>.

Zhang, Xiaomin; Song, Wei; Lui, Liming. “An Implementation Approach to Store GIS Spatial Data on NoSQL Database.” Paper presented at the 22nd International Conference on Geoinformatics. June 2014.