UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DEVELOPING AFFORDABLE SMART SOLUTIONS FOR POLICE REPORTING

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

RODRIGO ANTONIO COLLAO BENITEZ
Norman, Oklahoma
2018

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/301322

DEVELOPING AFFORDABLE SMART SOLUTIONS FOR POLICE REPORTING

A THESIS APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Joseph P. Havlicek

Dr. Ronald D. Barnes

Dr. Hazem Refai

©Copyright by RODRIGO ANTONIO COLLAO BENITEZ 2018
All Rights Reserved.

This thesis is dedicated to my parents, Jacqueline Benitez and Antonio Collao, for all their unconditional love and support.

Acknowledgements

I would like to thank Dr. Joseph P. Havlicek, chairman of my committee, for giving me the opportunity to join the CITS group and guiding me through my entire master's program. I thank him for all his ideas, suggestions, patience, answers to my uncountable questions, and for his willingness to help and support me whenever I needed it. I would like to also thank Dr. Ron Barnes and Dr. Hazem Refai, members of my committee, for their kindness and the selfless help they provided.

Additionally, I would like to thank Hesham Makhlouf for all the advice and support he provided during the development of this thesis and for having encouraged me to continue and complete this project together, just as we started it.

Finally, I would like to thank and dedicate this work to my mother and father, Jacqueline and Antonio, for their love, teachings and showing me that with hard work and dedication, I can achieve any goal. Thanks to my brothers Carlos and Fernanda, and my family and friends for all their support. And a special thanks to Mariana, for all her support, love, company and for keeping me inspired to continue during my master's program.

Table of Contents

Acknowledgements **iv**

List of Tables **viii**

List of Figures **ix**

Abstract **xi**

Chapter 1. Introduction **1**

Chapter 2. Background **3**

2.1 What are Electronic Citation (eCitation) Systems? 3

2.2 Importance of Accurate Data 4

2.3 Previous Approach 8

2.4 ECitation Software On The Market 9

2.4.1 TraCS 9

2.4.2 digiTICKET by Saltus 12

2.4.3 SceneDoc 16

2.4.4 Thin Blue Line 18

2.4.5 Quicket Solutions 20

2.4.6 Brazos Technology 22

2.4.7 PARIS by CITS 24

2.5 Mobile Apps 25

2.5.1 Web Approach 27

2.5.2 Hybrid Approach 28

2.5.3 Cross Compiled Approach 30

2.6 Touch Screens for User Interfaces 32

Chapter 3. An Affordable eCitation Application **34**
3.1 The Beginning of Electronic Data Entry in Oklahoma 34
3.1.1 The Oklahoma TraCS Pre-Pilot Project 36
3.1.2 Pre-Pilot Project Results 38
3.1.3 Official Oklahoma TraCS Pilot Project 39
3.1.4 Pilot Project Outcomes 41
3.2 Crash Reporting System (CRS) 42
3.3 PARIS, TraCS Successor 44
3.4 Data Flow 48
3.5 Problems with the Current Implementations 50
3.6 Potential Solution 51
3.6.1 Choosing a Framework 52
3.6.2 React Native 53
3.6.3 Native Script 55
3.6.4 Xamarin Forms 56
3.7 Printing Hardware 57
3.7.1 Choosing an Affordable Printer 59
3.8 Application Features 60
3.8.1 Secure Login 60
3.8.2 Barcode Scanning 60
3.8.3 Validation Rules 62
3.8.4 Add Multiple Violations 63
3.8.5 Crash Finder 64
3.8.6 Citation Generation 65
3.9 Application Back-end 65
3.10 Citation Manager 66
3.10.1 Client Side 66
3.10.2 Server Side Web API 67

Chapter 4. Implementation Details **68**
4.1 Xamarin Forms Overview 70
4.2 Portable Class Library 71
4.2.1 Dependency Services 72
4.3 Project Structure and Libraries Overview 74
4.3.1 Model-View-ViewModel Library (MVVM) 74

4.3.2 PARIS Plus App Library 77
4.3.3 ID Parser 77
4.3.4 Legacy Dropdowns 77
4.3.5 Location LookUp 79
4.3.6 Printer Library 80
4.3.7 User Control Library 81
4.4 User Interface Overview 81
4.5 Authentication 83
4.6 Barcode Scanning 87
4.7 Validation Rules 89
4.8 Multiple Violations and Citation Generation 91
4.9 Backend 92
4.10 Citation Manager 93
4.10.1 Overview 93
4.10.2 Data Acquisition 95
4.11 Budget 96

Chapter 5. Conclusion and Future Work **100**
5.1 Contribution 101
5.2 Future Work 102

Bibliography **104**

List of Tables

2.1 TraCS' partner states [58]. 12

4.1 Required budget to use PARIS+ including smartphone. 98
4.2 Required budget to use PARIS+. 99

List of Figures

2.1 Officer handwritting a citation on a roadway [42]. 9
2.2 Windows handheld device running Digiticket Software [68]. . . 13
2.3 SceneDoc app screenshot [69] 16
2.4 Thin Blue Line eCitation and eCrash [74]. 18
2.5 Quicket Solutions logo [63]. 21
2.6 Tyler Technologies logo [77]. 23
2.7 Web app block diagram, after [64]. 27
2.8 Hybrid app block diagram, after [64]. 29
2.9 Crossplatform app block diagram, after [64] 31

3.1 TraCS Pilot Project System Overview [23]. 39
3.2 TraCS Pilot Hardware Configuration [23]. 40
3.3 Current CRS login page render on an up-to-date version of Google Chrome Browser. 44
3.4 Collision report generated using SAFE-T [8]. 48
3.5 Oklahoma collision data flow [5, 24]. 49
3.6 Percentage of Stack overflow questions [66]. 54
3.7 Xamarin forms demo app screenshot from official website. . . . 56
3.8 Star thermal printer. 58
3.9 Arizona-U.S. driver license example. 61
3.10 Structure of bar code and symbol character [67]. 61
3.11 PARIS Screenshot showing its validations. 63
3.12 PARIS Screenshot showing the menu to add more violations or warnings. 64

4.1 Shared class example. 70
4.2 Bluetooth interface. 71
4.3 Architecture of Xamarin Cross-Platform using PCL [20]. . . . 72
4.4 Dependency Services overview. 73
4.5 Bluetooth interface implementation for Android. 74
4.6 Example solution (.sln) for a basic iOS and Android application. 75

4.7 PARIS+ libraries overview. 76
4.8 MVVM class diagram [52]. 76
4.9 Driver's license class usage example. 78
4.10 Dropdown usage example. 78
4.11 Location look-up library example code. 79
4.12 Printing library example. 80
4.13 Custom user control usage. 81
4.14 Authentication block diagram. 85
4.15 Authentication flow chart. 85
4.16 PARIS+ login page. 86
4.17 Barcode calling example. 87
4.18 Barcode scanning on the application. 88
4.19 Barcode scanner flow chart. 89
4.20 Screenshot showing validations on the form. 90
4.21 Screenshot of a rendered citation form. 91
4.22 Citation Manager sign-in page screenshot. 94
4.23 Citation Manager dashboard screenshot. 95
4.24 Citation Manager block diagram. 96

Abstract

DEVELOPING AFFORDABLE SMART SOLUTIONS FOR POLICE REPORTING

Rodrigo Antonio Collao Benitez, M.S.
The University of Oklahoma, 2018

Supervisor: Joseph P. Havlicek

Due to the high costs of many eCitation systems currently on the market, many police agencies are still using pen and paper to fill out their citations. This traditional citation method may be cost effective. However, it is time consuming and the chance of errors occurring on the form are high. PARIS, an eCitation system developed by The University Of Oklahoma (OU) Center For Intelligent Transportation Systems (CITS), has solved many of the issues faced by these agencies but because of the high cost of implementation, it is not a viable option for all police agencies in the State of Oklahoma. By taking advantage of the smartphones already carried by most officers and the help of Xamerin forms, I offer a solution based on the current design and implementation of PARIS called PARIS+. The development of the cross platform application, PARIS+, creates a police data collection system that is both more affordable and easier to use than currently available systems while still delivering high accuracy data for electronic citations. Since the system was also developed with scalability in mind, it has the potential to be able to support Crash, Driving Under Influence(DUI), or Stored Vehicle Report Forms in the future.

Chapter 1

Introduction

Even with all of the technology currently available to us, there are many police agencies that are still reporting crashes and speeding drivers using pen and paper [4, 32, 46]. This process is inefficient as officers take a long time to write these citations and having incorrect collected data is always a possibility as there are no validations systems that can help the officer while filling the form. Incorrect information can add to this already lengthy possess because the forms get rejected frequently as they are very likely contain errors because of the complex nature of these forms [31, 32, 47, 76].

It has been demonstrated that using technology the data collection process can be improved as providing systems that are simple to use and have strong validations can prevent the mistakes on forms [23, 26]. These systems besides increasing the accuracy of the data are also capable of reducing the amount of time needed to fill the form, as the number of keystrokes is kept dramatically reduced and officers can add violations with just one click instead of having to fill out an entirely new form [76]. Although current technologies have provided great improvements to the traditional handwritten method, a solution that can be afforded by all agencies does not exist in the market [4, 32]. The main focus of this thesis is in finding a way to provide a solution to this problem by developing a cross-platform application to create a police data col-

lection system that is less expensive and more simple to use than currently available systems while still delivering highly accurate data. I will first explain why it is important to have accurate data and discuss the current solutions in the market. Next, I will explain the solutions that The State of Oklahoma has implemented over the years to improve its data accuracy and I will mention all of the options for developing cross-platform applications, and explain how touch screens can be a feasible solution. All this to later propose my solution to this problem and explain the details of implementation which make this system possible.

In this thesis, I worked together with another student and developer, Hesham Makhlouf, and Dr. Joseph P. Havlicek, who with I shared and discussed many of the options and problems we faced during the development of this solution.

Chapter 2

Background

Over the last two decades, the emergence of electronic citation (eCitations) has positively impacted law enforcement by transforming their old handwritten forms into digital forms that can be filled easily and quickly with the least amount of errors [31]. All of this has been done with the help of eCitations while also increasing the effectiveness and safety of officers in the field.

2.1 What are Electronic Citation (eCitation) Systems?

According to [31], Electronic Citation (eCitation) are systems that allow police officers to write citations in a digital manner and electronically submit and print them. These systems, in addition to replacing the traditional handwritten citations, provide efficiency and productivity [33]. Having digital data entry allows for strong validation rules and ensures organized, legible and accurate citations which reduces errors and dismissals in court [26]. The eCitation system can capture, store and transmit data and make it available for multiple users in almost real-time [31]. All of these features allow the agencies to recapture revenue that is currently lost to errors common in handwritten tickets and also allows for the possibility of increased revenue since the issuing process is reduced in time [31].

The following list of features which are commonly found in eCitation

systems was given in [31]:

- Quickly capture driver's information by using barcode scanners. This data is used to automatically populate the forms
- Add extra warnings or violation with just one click
- Synchronize and submit citation data wirelessly to law enforcement agency's record management system (RMS) and any other government entity provided access to the system
- Strong validation rules that prevent the officer from making mistakes
- Receiving background information on the person being written up from third party software integration that accesses criminal data

2.2 Importance of Accurate Data

The book "The Role Of Data" [26] states that:

> "In today's complex, rapidly changing environment, the Secretary of Transportation, as chief adviser to the President and advocate before the Congress on national transportation policies, must have the capability to anticipate problems and devise policies to shape the transportation system of the future."

The authors of [26] say that this work requires plenty of data to measure performance and critical thinking in order to understand the data and transform it into a useful tool that will help make the best decisions. They also point

out that data not only helps in making decisions but also helps in creating the national policies that all agencies have to follow.

Also, [26] brings up an interesting point by mentioning how, in order to stay competitive, the CEOs of many large companies have access to different analysis tools which help them assess the data they have been able to collect. If private companies are able to obtain and track this information, the manager of a $30 billion *public* agency whose actions can affect a considerably large amount of people, goods, and services should also have the same capabilities.

Advanced applications for data collection in the transportation sector have the potential to enhance the speed and quality of data while simultaneously reducing its cost and the time it takes for reporting [26]. As mentioned in [75] an important aspect of data is for it to be uniform. This means that all of the data must follow the same format and structure. They also mention that if all of the advanced applications follow a uniform coding system, such as the one from the National Highway Traffic Safety Administration (NHTSA) for state accident reports, we would be better equipped to make smarter decisions in providing safer roadways. The authors also mention that there is a great amount of potential in the idea of using applications to assist police in their data recording. These applications would make it possible for officers to have on-site accident data recording tools.

The proper recording of traffic accidents and speeding data by police has always been an obstacle in the collection of meaningful state-level data [26]. Officers do their best to ensure that they enter data accurately according to the data manual [42]. However, they are still humans and these forms that they fill out can be time-consuming and complex. Thus, it is not uncommon

for mistakes to be made while filling out citations [31]. The authors of [26] also mention that the lack of efficient measuring tools that help in determining the position of the accident can also play a role in proper data recording. These authors believe that providing police with computers that allow them to enter accident and speeding data directly into the computer with set-up validation rules could noticeably improve the quality of data collected by the state and local police or other highway safety personnel.

We are constantly advancing our developed technologies so they can continue to offer improvements to our lives. These advancements do not only occur in data collection. Only a few years ago, the capabilities of phones were limited to calling. Today, the phones on the market must offer many more features in order to fit our current needs and expectations. Just like phones, the methods we use to collect and analyze data in a more cost-effective manner are constantly in need of improvement. Having the most up-to-date and accurate data is essential in making well-informed decisions. As mentioned earlier, in addition to being accurate, data must be uniform. No matter how accurate data is, it becomes much more difficult to work with if it is not uniform. As the authors of [75] mention, it is nearly impossible to accurately analyze non-uniform data. This is why they see the fact that different states and departments vary in the way they collect their data as a cause for concern. Although the authors consider this to be a very important issue, they express that it is difficult to address when not all agencies have access to the same technologies. To try to improve the situation, the Transportation Research Board (TRB) together with Transportation Asset Management, Statewide Transportation Data and Information Systems, and Geographic Information Science and Applications

hosted a peer exchange in order to bring both data and highway safety professionals together to discuss the most important features in integrating roadway, traffic, and crash data sources to support safety management. This discussion illustrates that, while some agencies have been trying to reach data integration capabilities, many others have indicated that significant issues remain and must be addressed before they can be integrated [75].

As you can imagine, data alone does not provide enough information; multiple levels of analysis are required to translate data into useful information for policymakers. This is another point that the authors of [26] focus on. They also mention that safety is one of the main goals of transportation engineers. As mentioned earlier, government agencies spend a lot of time and effort analyzing traffic accidents in order to find some correlation between them. This information could potentially be useful in providing better and safer road conditions in the areas where they are needed [3]. As you can see in [9, 57, 80] not only are government agencies trying to analyze this data, but there is a great deal of research being done on this topic in order to help find better prediction methods which could potentially help in preventing future accidents. So we can conclude that, in order for this data to be useful in increasing safety, it is necessary to have enough accurate data. Obtaining accurate data can potentially be made easier to achieve with the use of technology in the field, where the data is being collected. This technology can ensure the forms are filled correctly and the data is as accurate as possible.

2.3 Previous Approach

Before eCitations, as mentioned in [76], officers used to write citations with pen and paper which caused many delays in the submissions of their forms. Not only did this approach cause delays, but the process was a frustrating experience for the officers who had to spend many hours filling out these forms by hand and then waiting to have them reviewed and accepted [46]. Additionally, handwritten citations require a lot of time and training on the part of the officers in order for them to fully understand how to properly fill out the forms [46]. Handwriting a ticket may not seem like a difficult task, but when taking into consideration the other factors that an officer is dealing with, the obstacles become more clear [42]. While concentrating on writing tickets, officers are standing in the middle of the road, could be in a hurry because they may have just received another call, and they must constantly be aware of the actions and intentions of the driver they pulled over [42]. It is difficult task despite the training. Because of all of the added situational factors and distractions, it is likely and understandable that even the best officers will eventually make a mistake on the forms [46]. Jim Nielsen [42], a police officer and professor of the criminal justice college degree at Rasmussen College, writes in his blog about how he needs to make many decisions in a matter of seconds and how, in these circumstances, every detail is important. He calls this "The Art of Law Enforcement". By this, he means that managing these circumstances and balancing all of these distractions requires a strong attention to detail. Unfortunately, this pressure can lead to mistakes being made. These mistakes made by officers, in turn, could lead to a higher chance of dismissal in court [76]. These mistakes are so prevalent that a quick online search about how to get



Figure 2.1: Officer handwritting a citation on a roadway [42].

a ticket dismissed will bring up many results suggesting that the simplest way is by finding a mistake on the form. There is even a startup, Fixed [37], that provides a professional service for getting tickets dismissed in court by finding errors on the ticket. As the authors of [76] mention, to avoid these dismissals, the government has been trying to improve this process. In many instances, the government has even tried to encourage the use of applications that could help the officers collect accurate data in the field by using device sensors and making that data accessible almost immediately to other interested parties.

2.4 ECitation Software On The Market

2.4.1 TraCS

As explained in [54], TraCS stands for *Traffic and Criminal Software*. TraCS is a data collection and reporting software program developed by a contortion of American states and Canadian provinces with funding from the United States federal government. As the author explains, the goal of TraCS is to automate the capture and transfer of incident data in the field. It is administrated by

and receives ongoing support and upgrades from a coalition of users called the National Model for the Statewide Application of Data Collection and Management Technology to improve Highway Safety.

The National Model

The National Model for the Application of Data Collection and Management Technology to Improve Public Safty, an initiative by the Iowa Department of Transportation, was the project that launched TraCS. The National Model was founded by a consortium of 18 US states and Canadian provinces that are designated as TraCS states and provinces [23].

The consortium uses TraCS to enable local, state, and federal public safety entities to share information [54]. Shared use of this common software allows the establishment of best practices to ensure accurate data acquisition.

Meyer [54] gives a chronological background of the history of TraCS that is summarized here to provide a better understanding of the history and current status of TraCS.

- In 1994, the Iowa DOT (Department of Transportation) worked with the law enforcement community to develop a crash reporting system in order to capture data on crashes. One year later, they decided to add traffic citations and commercial motor vehicle inspections. The Federal Highway Administration saw the potential in this software and, in 1996, funded the State of Iowa to share TraCS with other states.

- By 2000, TraCS was growing and features such as the Incident Location Tool (ILT) were being added to collect GIS coordinates to locate crashes

and incidents. (GIS stands for geographic information system. It is a framework for gathering, managing and analyzing data. It analyzes spatial location and organizes layers of information into visualizations using maps and 3D scenes.) In this same year, TraCS also released the first development kit to allow states to manage the evolution of their current paper forms into TraCS electronic forms and customize them to their needs.

- In 2011, a browser-based version of TraCS was released that targeted rural agencies which could utilize TraCS with minimal support costs.
- The last reported feature by [54] was a TraCS RMS that was released in the 2nd quarter of 2014.

TraCS' driver's license bar code scanners could save officers on the streets a great deal of time according to the authors of [76]. These authors also estimate that using bar code scanning devices to populate electronic reports can eliminate up to 750 keystrokes in a crash reported and up to 200 keystrokes per traffic citation which would reduce the overall time of writing the citation.

In [76] the authors also claimed that TraCS reduced the time it took for a crash report to complete a full cycle from up to 18 months in the State of Iowa to 18 hours by transmitting data directly to all interested parties.

The states that currently use TraCS based on The National Model website [58] are the listed on table 2.1.

Every state and agency using TraCS must train some of its officers to maintain their version of the system and adapt it to their needs. Each state

Alaska	Arizona	Florida	Illinois
Iowa	Manitoba	Minnesota	Nebraska
New Mexico	New York	North Carolina	North Dakota
Pennsylvania	South Dakota	Vermont	Wisconsin

Table 2.1: TraCS' partner states [58].

needs to develop their own versions of TraCS police forms, know as Tracs Pack [23]. While this could be a useful feature to have, it is expensive and difficult for agencies to maintain [5]. Eventually, the cost and maintenance required by the system and the need for features not included in TraCS caused the State of Oklahoma to opt out of the TraCS program in favor of a more flexible and easy to modify system [5, 32, 46].

2.4.2 digiTICKET by Saltus

Saltus provides an eCitation solution called digiTICKET designed for state and local governments. According to the website [68], they focus on ease-of-use. Saltus expresses that the digiTICKET eCitation solution is not just an electronic ticket book. It also provides other potentially useful features such as photo captures, GPS mapping and enhanced reporting.

According to Saltus' description, to run digiTICKET, a Windows Mobile device or any tablet or laptop running Windows XP/7/8/10 is needed. This feature allows agencies to utilize their current investments in hardware.

Some of the benefits listed by digiTICKET include the following:

- Improved officer safety
- Higher productivity



Figure 2.2: Windows handheld device running Digiticket Software [68].

- Increased revenue by reducing ticket errors
- Automation of the ticketing process

How Does It Work?

According to digiTICKET's documentation, this is how the process of writing a citation works [68]:

1. Scan a driver license

   By using digiTICKET, an officer can scan the back of virtually any states drivers license or ID card to enter all of the driver's information into the ticket automatically whether the driver's license stores the data on a

magnetic strip or a barcode. The digiTicket system also offers the optional capability to perform queries to The National Crime Information Center (NCIC) on people and vehicles or pull query data from other mobile software applications. If this capability is implemented, the returned query data can be used to automatically fill person and vehicle fields on the ticket.

2. Enter other ticket data

   Using drop-down lists and easy-to-use screens, all data required to complete a ticket can easily be entered in less than 60 seconds.

3. Add Violations

   Most common violations are added by selecting from a drop-down list or a simple key word search of the violations description no more cheat sheets required. Officers can also add warnings and flag violations as "grant" tickets.

4. Print the ticket

   Paper tickets are printed from an affordable Bluetooth thermal printer. Up to eight violations can be printed on a single four-inch wide ticket, formatted to look just like the pre-printed ticket books. Officers can even use digiTICKET to print differently formatted tickets based on ticket type (i.e. parking, traffic or code enforcement).

5. Capture photo, electronic signature, voice note or fingerprint

   Officers have the option of using the digiTICKET handheld ticket writer to capture the violators signature along with a digital photo. If comments

require a long narrative, the officer can also capture a voice recording of the stop.

6. Synchronize ticket to the web application

   Officers can easily submit ticket data to the digiTICKET web based software from their handhelds, laptops or tablets running the digiTICKET mobile software.

7. Manage Tickets

   Once on the web software, designated users with web access can access tickets online. Tickets can be reviewed and approved, edited, reproduced in PDF format, and exported to the Court or Record Management System. This is not explained on their website but is very likely that this will be hosted on digiTICKET servers, and it is not clear but very likely that the data will owned by digiTICKET.

8. Manage statutes and court dates

   Users with appropriate permissions can add, edit, or remove statutes and update court dates and times. Court dates can be set to advance automatically to the next available date based on docket size or number of days to the next appropriate court type.

9. Create reports

   All ticket data is stored on the digiTICKET web application and can be used to create reports that the RMS or Court system does not already provide.

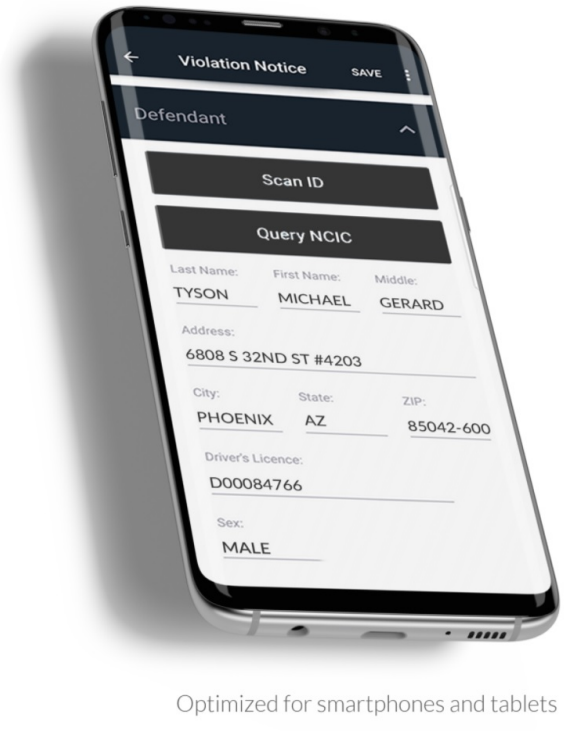


Figure 2.3: SceneDoc app screenshot [69]

.

2.4.3 SceneDoc

On its website [69], SceneDocs claims to be the first mobile tool in public safety. The website also claims that they are trying to redefine the role of mobile technology in public safety.

Some of the benefits that SceneDoc claim include the following:

- Proven to reduce stop time more than 50%
- Re-capture revenue previously lost to handwriting errors
- No more manual entry into court systems
- Less time spent on the side of the road

- Issue a citation using any smart device
- Built-in ID scanner populates fields automatically
- Issue violator copy using bluetooth printer
- Data immediately flows to the cloud where it is now searchable
- Data is ready for direct export to the courts

How Does It Work?

SceneDoc describes on its website [69] that it was built while keeping configuration in mind. According to their website, SceneDoc was designed to offer a smooth user experience and to enable the officers to easily insert the data that they collect in the field. For officers to start using SceneDocs, it is as simple as having the app installed on a portable device such as a smartphone. Unlike other competitors, SceneDocs takes advantage of the mobile devices that most officers currently have in their possession and makes use of them in order to collect valuable data.

SceneDoc listed features include the following:

1. Work from any device
2. Functions online or offline
3. Real-time sync to web portal
4. Data exports to existing systems
5. Built-in efficiencies like ID scanner and voice dictation



Figure 2.4: Thin Blue Line eCitation and eCrash [74].

2.4.4 Thin Blue Line

The Thin Blue Line (TBL) eCitation system, according to their website [74], provides a platform that generates digital citations using an iPhone or iPad. The software TBL provides can be deployed with minimal effort from a law enforcement agency. In most cases, the agencies have to provide only a scanned citation format and TBL claims that they will do all of the required setup. The app is distributed using the Apple App Store for iOS devices.

TBL citations module, according to their documentation [74], is pretty simple to use and requires almost no training. Their module comes with an ID scanner that can scan everything from licenses to international ID's and immediately populate the information in the required fields. They also provide auto-complete drop-down menus and simple search queries that allow for a relatively straightforward user experience. Some fields, such as the court citation and date fields, are even automatically populated as soon as a new citation is created. On their website, they explain that they added this feature in order to avoid errors from the officers in the field and to save them the time it would take to enter this information. One of the unique features offered by TBL is

that a citation can be generated in less than 60 seconds, and with advanced users, it can even be done in less than 30 seconds [74].

Costs

Since TBL is using the smartphone approach (only supporting iOS devices like iPhones and iPads), their system has the advantage of having basically no hardware costs as most officers already have these devices [74]. Another positive aspect of TBL is that deployment becomes quite simple thanks to the app store provided by Apple. This is especially useful since iOS devices from Apple are quickly becoming standard in most industries across the U.S as they are liked for the security they provide and their hardware capabilities. Additionally, Apple is continuing to increase its popularity by targeting educational businesses by offering more affordable models of their products [11, 12, 19, 22, 53]. As an extra feature, Thin Blue Line provides lifetime updates, upgrades and ongoing services to all of their customers.

Modularity

Another important feature TBL mentions [74] is that no other competitor offers its modular design. Their universal reporting system comes with different modules that can be added based on the agency's need. They claim that this allows for customization that can potentially improve user experience and increase user satisfaction. The modules described by TBL include the following [74]:

- Citations
- Crash Reporting

- Field Interviews
- Tow Forms
- Reporting

Compliance

TBL claims [74] that an element that makes their system a viable option for agencies is the compliance aspect. Because their eCitation is compliant with The Criminal Justice Information Services Division, a division of the United States Federal Bureau of Investigation (FBI-CJIS), National Incident-Based Reporting System (NIBRS) and Health Insurance Portability and Accountability Act (HIPAA), it allows the system to be used for Civil, Traffic, Parking, and Criminal Citations. As mentioned previously in this chapter, having uniform data is crucial to having reliable data. The fact that the data is uniform allows it to be used and shared seamlessly across different organizations [74].

2.4.5 Quicket Solutions

As they explain on their website [63], Quicket solutions is a cloud-based e-citation system. The description of Quicket as an eCitation System explains that, like almost all of the previous approaches, it allows for the recapture of lost revenue and increases officers' safety.

The benefits that Quicket Solution claims to provide include the following [63]:

- *Zero Upfront Costs:* For a monthly fee, they offer the whole solution which provides software, cloud service and support. There are no upfront



Figure 2.5: Quicket Solutions logo [63].

customization or installation fees.

- *Cloud-Based:* Since they provide a cloud based service, agencies are not required to maintain their own servers. This can be both an advantage and disadvantage. It is an advantage because they do not need to have the required IT infrastructure. However, there is a high chance that the data will not be owned by the agencies but by the provider.

- *Data Analytics:* Quicket Services provides powerful data analysis using data mining. Thus, allowing the agencies to detect patterns and make predictions.

- *Robust Security:* Quicket Services is in compliance with the Criminal

Justice Information Services (CJIS) division of the FBI.

Products

As they explain on their website, Quicket's E-citation module is a Windows and Android compatible application. Some of the features they offer include the following [63]:

- Integration with courthouses
- Integration with Quicket cloud
- Drop-downs to prevent errors in typing
- Auto population of fields

Quicket also provides an Incident and Crash platform that allows users to have access to the following [63]:

- In-depth reporting
- Digital signatures
- Live collaboration between officers and supervisors
- Artificial intelligence to link data

2.4.6 Brazos Technology

Tyler's Brazos e-citation software solution provides a mobile electronic citation solution [77]. It provides flexibility for the agencies with no required changes to their current infrastructure [77].



Figure 2.6: Tyler Technologies logo [77].

According to Tyler services, officers can submit their citations using a mobile device that automatically uploads the data into any system of the officer's agency [77]. The key features mentioned by Tyler Technologies about their Brazos e-citation include the following [77]:

- Reduces time spent on traffic stops and parking citations
- Increases officer and violator safety
- Eliminates data entry errors on citations
- Eliminates data entry into court and public safety records systems
- Reduces total cost of processing citations
- Requires minimal IT support
- Provides complete citation, statistical and mapping reports
- Auto-fills suspect information from drivers license, VIN number or registration sticker
- Operates on any Microsoft OS mobile device
- Interfaces with any public safety records management or court case management system

2.4.7 PARIS by CITS

Police Automated Record Import System (PARIS) is a record management system developed by the Center For Intelligent Transportation Systems at The University of Oklahoma [46] that enables the Oklahoma Department of Public Safety to electronically import crash data from a large number of agencies into existing DB2 data warehouses. PARIS allows agencies to import crash data while also avoiding undesirable costs and security problems. Not only this, but PARIS also provides validated and accurate data and reduces the submission time of data [47].

Some advantages of PARIS include the following:

- Reduction of time spent on a traffic stop
- Increased safety for both officer and violator
- Strong validation rules both on the clients and the servers. (Client validation means the system will prevent the user from submitting data that is not complete or incorrect, like incorrect date format, a field with more characters than the maximum length. Server side validation will make sure the data is correct before actually saving it to the databases.)
- Reports and statistics
- Interface with court management system
- One interface for all applications. On previous approaches, officers had to move between applications to find the information they needed

- Offline login. This allows officers to generate forms even when they do not have internet coverage

PARIS also provides its users with a sophisticated *crash finder*, a tool that allows the officers to retrieve all of the location information for the crash forms by simply clicking the location of the incident on a map [32]. PARIS also allows the officers to draw an accident diagram using professional tools and then import the drawing back into the application and attach it to the crash report forms.

PARIS has an ID scanner that not only auto-populates the information of the form but also retrieves background information about the driver [32]. This way, officers can have a better understanding of what kind of person they are dealing and whenever that person has previous violations and/or outstanding warrants or charges.

2.5 Mobile Apps

Nowadays, almost all working adults have a smartphone. They are no longer considered a luxury item [38, 51, 62]. We are in an era of mobility- the growth of mobile markets in terms of both the number of devices sold (smartphones and tablets) and the number of downloaded mobile applications illustrates this [14,40,49,73]. As mentioned in [49], the growth of mobile smartphones also comes with the fragmentation of mobile platforms (or mobile OS) which makes developing applications much more difficult and time consuming. Applications must be available on as many platforms as possible to be competitive. Luckily, as has been seen in the past few years, platforms have been appearing and dis-

appearing and have now reached a point at which only two leading platforms remain and dominate the mobile device marketplace. The most recent example of this is Windows Phone OS, a response from Microsoft to Android, and iOS. However, they ended up canceling the program last year [44, 79]. Every platform comes with an Software Development Kit (SDK) and each requires specific programming skills in terms of both languages and API so that developers can code applications on their platforms [13, 21, 39, 56]. This creates additional redevelopment costs for software developers who who must work on multiple platforms. Because of this, plenty of research has been invested into trying to develop a way to write code that can run on all platforms [49].

Consider all of the smartphones running Android; there are many different manufacturers such as Samsung, Huawei, HTC, Google, Motorola, and all of them provide several phone models that have different specifications like screen size, available memory, and processor speed. There are many different types of smartphones being used. Because of the many variations offered, modern applications are expected to run on a variety of web and mobile platforms with diverse software and hardware level features.

Applications may require substantial effort and re-development to port from one platform to another. This dramatically increases the required time and skills needed by the developers and also increases the cost of production. A solution is represented by a framework for cross-platform development [49]. Raj and Tolety [49] classify cross-platform frameworks in four categories: web, hybrid, interpreted and cross-compiled.

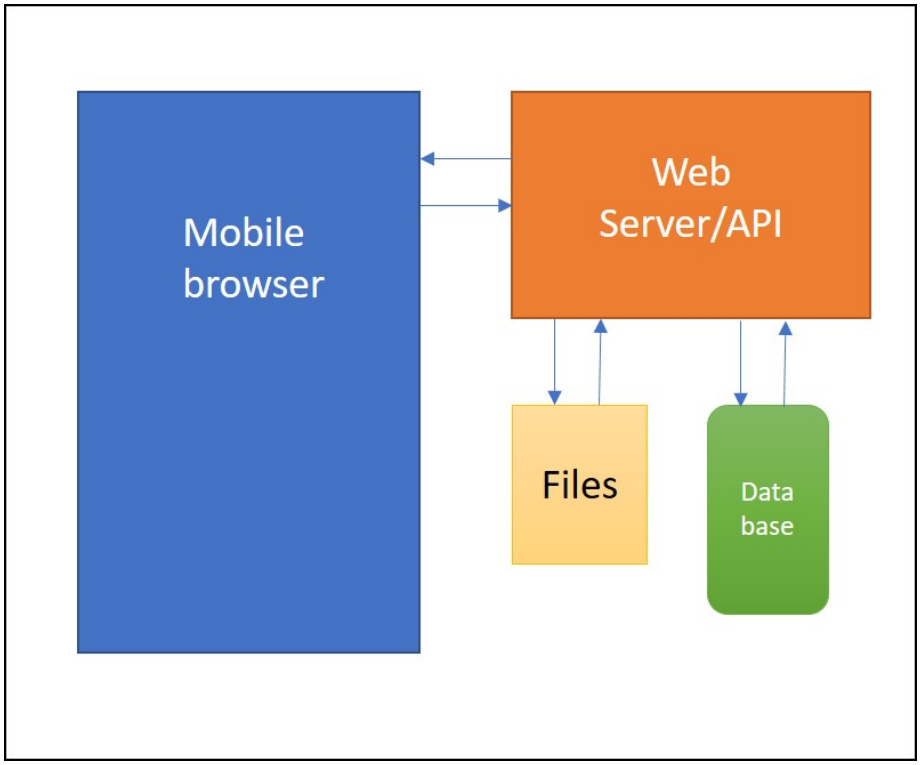


Figure 2.7: Web app block diagram, after [64].

2.5.1 Web Approach

As explained in [49] the web approach is a website that behaves as an application but is designed to execute in the web browser of a mobile device. Usually, these are developed using HTML, CSS and JavaScript. There is no need to have any application installed; the only application needed is the browser and the server-driven web application data. Since the application is based on the browser, it is platform independent. Figure2.7 explains the common flow of a web application. A mobile browser talks to a server which replies with files and records from the database that then become rendered on the browser.

The advantages mentioned by Raj and Tolety are as follows [64]:

- It does not require installation on the devices- the user simply needs to navigate to the url.
- Since all of the data is hosted on a server, updates to the system are maintenance free.
- All web browsers work parsing html. Therefore, one web user interface can be reused on any platform.

The disadvantages are as follows [64]:

- The web platform cannot be distributed on any app store. This could negatively impact the distribution of the system since it is not easily available on the app store.
- The performance might not be as good as the native app, especially if there is a bad connection.
- The native capabilities on the devices such as specific hardware like sensors (accelerometers, fingerprint reader) cannot be accessed.
- Developers do not have control of how the browser renders the view.
- It has to have a working internet connection to function. If the network is down, the app cannot function.

2.5.2 Hybrid Approach

According to [49], Hybrid apps are in between web and native technologies. These apps run on an embedded web container on a mobile device. Basically, it is the native web browser of the operating system running in full screen with

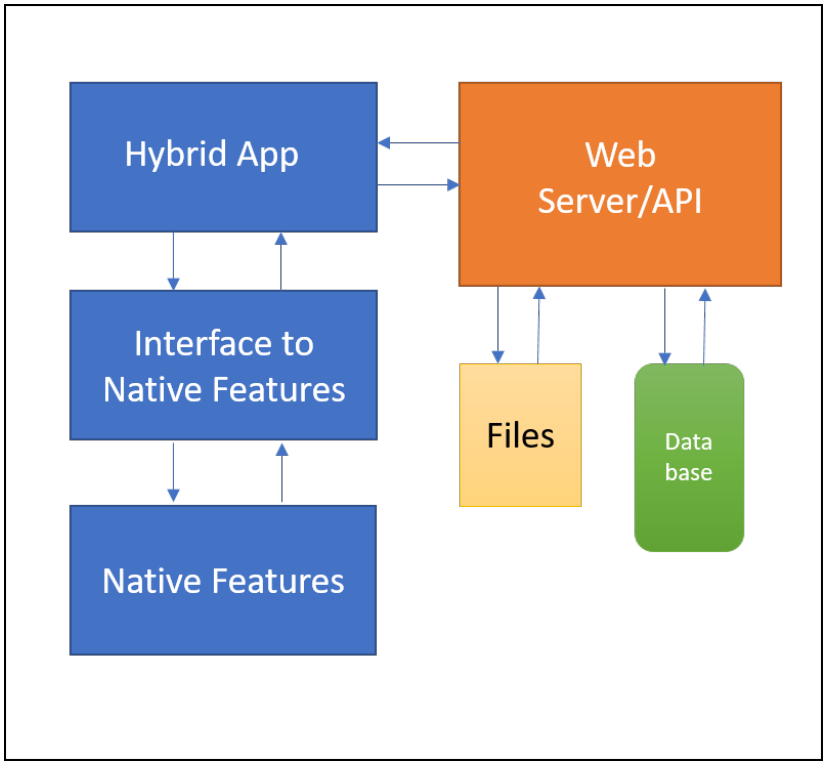


Figure 2.8: Hybrid app block diagram, after [64].

all of the static information loaded on the phone so it does not depend on a web server all the time (unless it needs to consume data). See Figure 2.8. Besides this, as the author of [49] mentions, it is very similar to the web approach; some native capabilities are still accessible through an abstraction layer- usually javascript APIs- that allow the developer access the native features. Hybrid apps can work both as consuming services and as standalone applications. Since hybrid apps can also work standalone, they need to be downloaded and can be distributed using app stores.

The advantages of using hybrid apps, as mentioned in [49], are as follows:

- It is available to download on the app store.

- User interface is totally reusable across all different platforms.
- Access to native features is available through an abstraction layer.
- It is available for offline use and is powered by the device capabilities.

The disadvantages of having hybrid apps, as mentioned in [49], are as follows:

- Performance is not as good as native applications since it is still running through a browser.
- Not all native features might be available.
- The user interface might not be as smooth as on a native app.

2.5.3 Cross Compiled Approach

From [64], we can see that when developing cross compiled applications, the cross compiler converts the source code to native binaries. All of this is handled on the back-end by a cross compiler that generates the executable code for each particular platform. See Figure 2.9. And, as the author explains, this is highly dependent upon the efficiency and reliability of the cross compiler.

Some of the advantages of having a cross compiled app are as follows [49]:

- They have all the native features possessed by any native app.
- All native user interface components can be used.
- Performance is very high compared to any of the previous approaches mentioned.

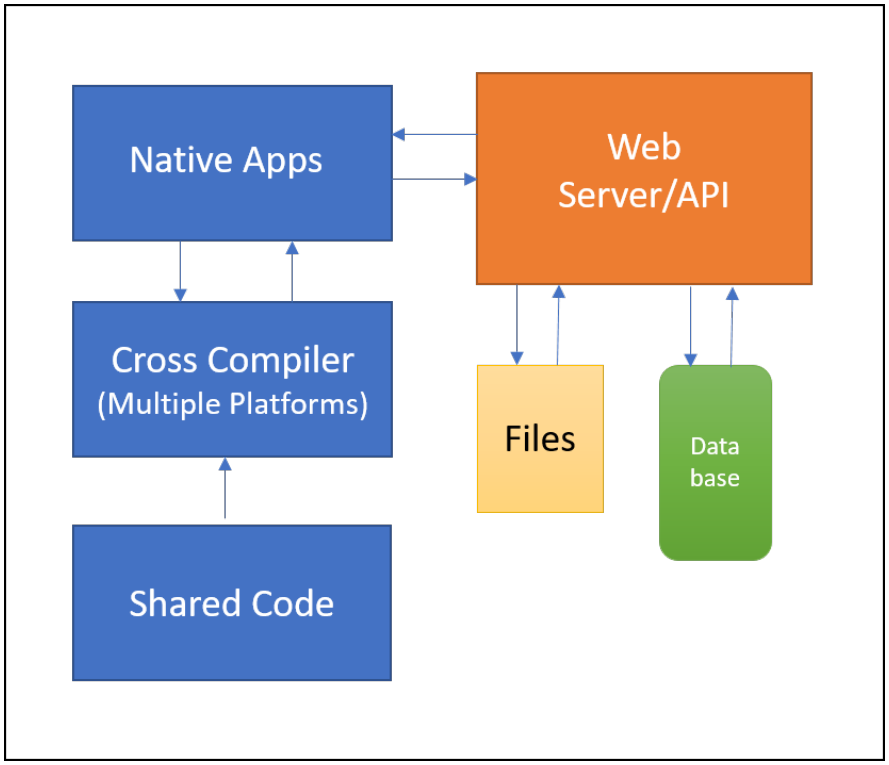


Figure 2.9: Crossplatform app block diagram, after [64]

.

Some of the disadvantages of using the cross compiled approach are as follows [49]:

- Some platform specific code might be needed as some parts of the interface might not be reusable.
- It works well for small applications. However, on more complex applications, native approach will have better performance than the cross compiled.
- It takes a longer time to compile.

Nowadays, the times when a native app is really necessary are decreas-

ing. With the current improvements in technologies, even web apps will start to have access to native features on the phone such as fingerprint readers [78]. Hybrid apps' abstraction layers to access native features are improving and the performance of these functions have increased noticeably [29]. We are arriving at a point at which native app development does not have to occur too often and is being left for only specific cases where its development is really needed [28].

2.6 Touch Screens for User Interfaces

Years ago, we started out with resistive touchscreen interfaces that were not considered to be user-friendly; these interfaces forced the user to use a pointing tool to be able to make selections on the screen. After a while, we moved on to the use of capacitive touch screens with multi-touch capabilities. Touch screens have been improved significantly in the last few years. These types of touch screens can currently be found on almost all consumer devices on the market. According to [6], capacitive touch screens are precise and eliminate the need for a pointing tool. This provides for a better user experience as the user is now able to use his or her finger to interact with the screen [6].

Albinsson and Zhai [6] explain that touch screens may not initially appear to be the most user friendly solution to user input. However, when considering the fact that both the data rendering and the interaction happen on the same surface, this makes touch screens one of the most direct user interfaces [6]. Albinsson and Zhai also point out that the fact that there is zero displacement between the user input and the system output. This feature makes touch screens one of the most intuitive interfaces to use. Albinsson and

Zhai goes on to explain that, since touchscreens have their control on top of the screen, there is no need for an extra input device. This can make them more robust than any other input devices such as a computer mouse. However, they can have some limitations. When the user uses their finger to interact with the device, their finger might cover part of the screen. Plus, using the finger as a pointing device might be difficult when the targets are smaller than the finger's width [6].

In [50] Lee and Zhai point out one important feature that touch screen interfaces have: they allow developers to place "soft buttons" based on their need. These buttons are rendered graphically and can appear or disappear based on the interaction context. They can also change in color and size based on their position and the screen space availability.

Touch interfaces have been revolutionizing many fields including aircraft. Avsar, Fischer and Rodden [15] show this while designing a novel touch screen interface to improve flight deck operations. They came up with promising results when they built an interface based on the touch input experience of the pilots.

There have been some studies done on trying to demonstrate the performance capabilities of touchscreen interfaces [25, 43, 50, 60, 70]. While they can have some disadvantages, depending on the application, touchscreens can be as fast as any other user input mechanism or even faster. This can happen as long as a good interface with big enough touch inputs is designed to take advantage of the touch capabilities of the screen [70].

Chapter 3

An Affordable eCitation Application

Before 2007, police officers in the State of Oklahoma were still writing citations and crash forms using pen and paper [5]. According to a contract [47] found in The Center For Intelligent Transportation Systems' (CITS or ITS) records, during this time, the Oklahoma Department of Public Safety (DPS) was receiving a large volume of traffic records from numerous police agencies across the state and maintaining a repository of records for crash reports, traffic citations, and others. Once the officers had written these forms, clerk-typists would manually enter the data for archival storage in the DPS main DB2 data warehouse where the forms were also scanned to create an electronic image for archival storage .

3.1 The Beginning of Electronic Data Entry in Oklahoma

According to a technical report written by the CITS group [23], there was a strong desire from federal and State levels to implement electronic police reporting with data validation, but the need for improved data accuracy and timeliness in Oklahoma increased notably subsequent to January 1, 2007, as the new Oklahoma Uniform Crash Report form was deployed. Following the Crash Form's deployment, the State of Oklahoma was experiencing difficulties with the automated scanning and data validation associated with it.

The Oklahoma Department of Public Safety (DPS) has been involved with TraCS since the earliest days of the National Model. Currently, DPS is the Oklahoma lead TraCS agency. However, sufficient funds were not available to support TraCS development as the report [23] explains. Before mid-2005, Oklahoma Highway Safety Office (OHSO) obtained funds for TraCS development and OHSO contracted the University of Oklahoma Intelligent Transportation Systems (CITS) to initiate efforts to do the following [23]:

- Develop the Oklahoma TraCS electronic forms suite
- Investigate options for the required infrastructure to support TraCS deployment to Oklahoma Highway Patrol (OHP) on a statewide basis
- Provide technical recommendations about strategies to deploy TraCS in Oklahoma

CITS then sent a team of developers to Florida Sate University in Tallahassee, Florida, to visit a TraCS laboratory to learn how the Florida TraCS Pack (as explained in the previous chapter this is the state's custom TraCS electronic forms suite) was designed and implemented [5].

This project was referred to as the TraCS Pre-Pilot Project and it continued through the end of 2007 [23]. As the report details, the Pre-Pilot project involved several contributions from a large number of individuals from several agencies listed on the CITS report, and members of DPS and OHP that held significant leadership roles.

3.1.1 The Oklahoma TraCS Pre-Pilot Project

The project begun on May 1, 2005, and, as the CITS report [23] explains, the goals of the project included the following:

- Development of the Oklahoma specific state suite of electronic TraCS police forms (also referred as the Oklahoma TraCS Pack). These forms included the following:

  1. The new Oklahoma Crash Report Form as the top priority.
  2. Oklahoma Citation Form.
  3. Oklahoma Contact Form, Felony Case Record Form.
  4. Financial Responsibility Form, Stored Vehicle Report Form.
  5. Several specialized versions of 7-Day Activity Report Forms.

- Evaluation of the technologies that could be used for the deployment of TraCS Pilot project for OHP, including hardware and software elements, and communications infrastructure. It was important to consider the cost of long-term support and maintenance and ease of achieving interpretability with existing police tools like Aspen(a commercial vehicle enforcement software) and MobileCop(police secure communication tool) while making the evaluations.

- Hardware and Software technologies recommendations for the official OHP TraCS Pilot Project deployment

At that time, the TraCS Pre-Pilot project became a major project for the CITS group [5, 24]. During the Pre-Pilot project, the CITS group ran several extensive evaluations on hardware and software to find out which were the

most suitable for the official project. The hardware evaluations included computers, peripheral devices, USB Hubs, external hard drives, signature pads, 2-D barcode scanners, magnetic strip readers, printers and communication hardware. During the software evaluations, the team tested the available Crash Diagram Tools, the options for TraCS database selection, interoperability with DB2(database product by IBM), Mobilis and INDICIUS (optical character recognition (OCR) software to import data from PDF documents), interoperability with MobileCop and Aspen(a commercial vehicle enforcement software) and MobileCop(police secure communication tool).

One of the most remarkable evaluations performed by CITS was the Communication Hardware. As explained in the report, CITS was able to *develop a capability for End Shifting TraCS forms over a cellular telephone circuit switched data link* [23]. The term "End Shift" originated from a legacy notion that Troopers/Police Officers would transfer forms at the end of a shift [32]. With the current TraCS technology of that time and with the appropriate designed communication network, it was possible for a Trooper to continuously End Shift electronic forms [23]. However, the CITS report also explains that TraCS provided only the software support and data protocols for the End Shift operation, but not any design or specification for the actual physical communication network. Since the hardware and infrastructure for each state is vastly different, the physical communication network for the End Shifts must be implemented and designed individually by each state.

Once a form or group of forms has been End Shifted, they become visible to the Trooper's supervisors using a desktop computer running a version of TraCS configured for supervisory functions [23]. Supervisors will then have

to access the forms and accept the forms or reject them if they contained any problems. If forms were accepted, they would then be transfered to a TraCS Server. However, if they were rejected, they would be returned to the trooper with a message explaining why it was rejected(this is called Start Shift operation as the officers receive rejected forms at the beginning of the next start shift). The trooper would then have to fix the error and submit it again and wait for approval. According to the report [23], in most states, these traffic records data must be extracted from TraCS Server database and transfered to other databases and records management systems. These extractions and transfer mechanism are not part of TraCS and they must be designed and implemented by each state.

3.1.2 Pre-Pilot Project Results

The Pre-Pilot project ran from May 1, 2005, to December 31, 2007. The necessary suite of Oklahoma forms (TraCS Pack) was developed including both new and previous versions of Crash Forms, Citation, Contact, Stored Vehicle Report, Financial Responsibility, Felony Case Record and Seven-Day Activity report forms. CITS also developed databases and validations rules for all of the forms. The necessary TraCS server was deployed at DPH Headquarters in Oklahoma City and the *Start Shift* and *End shift* were implemented using both, aircard wireless links and cellular circuit switched data links. A wide variety of peripheral devices were tested by the Pre-Pilot Troopers along with different crash diagram tools. CITS demonstrated the feasibility of deploying TraCS on an agency-wide basis. During this Pre-Pilot project, real crash forms were submitted electronically by the Trooper, Troy Thompson.

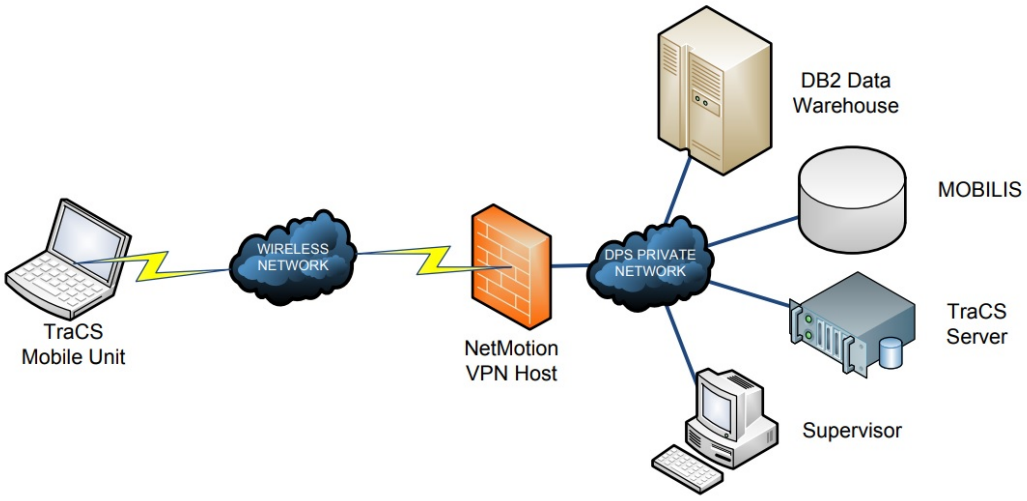


Figure 3.1: TraCS Pilot Project System Overview [23].

3.1.3 Official Oklahoma TraCS Pilot Project

The TraCS Pilot Project was based on the results of the Pre-Pilot. It initiated on January 1, 2008, and ended up lasting for one month [23]. Each of the Troopers was equipped with a laptop computer running Windows operating system with TraCS installed (TraCS Mobile Unit on 3.1). Each Trooper had to connect to DPS Virtual Private Network (VPN) using a software called NetMotion that was loaded on their computers. The internet connectivity was provided by GPRS aircards and TraCS servers were running on DPS Headquarters protected by the DPS private network firewall as can be seen in Figure 3.1. Supervisors used desktop computers, generally located at the Troop Headquarters, with TraCS installed. The approved crash forms were exported to the DB2 data warehouse and images of the crash forms where saved to the Mobilis document management system.

The hardware configuration was the same for all Pilot project Troopers

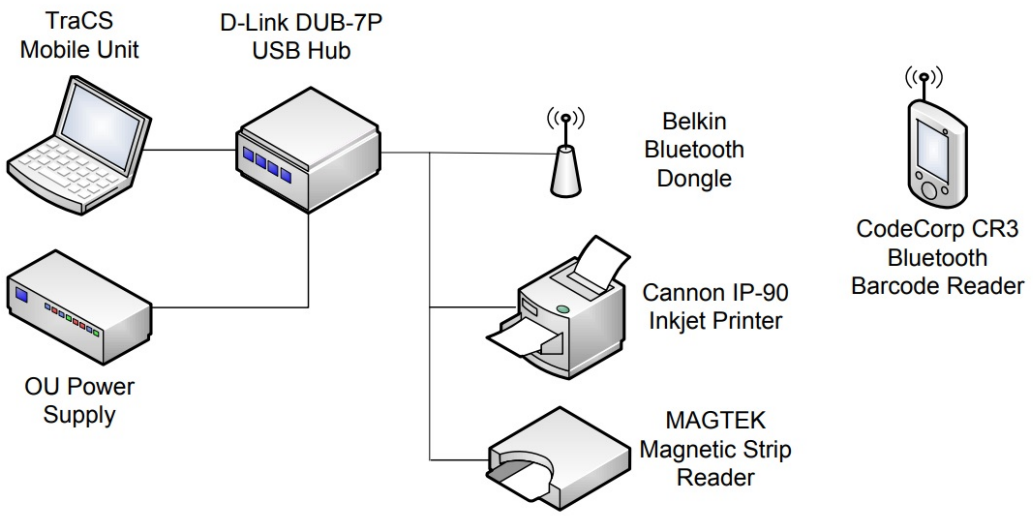


Figure 3.2: TraCS Pilot Hardware Configuration [23].

and, as mentioned previously, it was based on the results from the Pre-Pilot project. The computers were plugged to a USB Hub to connect all the peripherals and were provided with a power supply developed by the CITS group. A bluetooth dongle(for a wireless bluetooth barcode scanner), thermal printer, and a magnetic strip reader were attached to the USB Hub to interface with the computer. See Figure 3.2.

The software configuration had a COM Port Management service that allowed all the applications(TraCS, ASPEN, and MobileCop) to capture and convert data received on a physical serial port. The data was captured and then relayed to all the virtual communication ports on the machine monitored by applications such as TraCS, MobileCop or ASPEN.

3.1.4 Pilot Project Outcomes

The pilot project was conducted during the month of January in 2008. During the Pilot project, according to CITS report, nine crash reports were End Shifted by the Pilot Troopers and a total of 31 Crash forms were End Shifted during the Pre-Pilot and Pilot projects. Nine of the End Shifted Crash forms were approved and successfully scanned into INDICIUS. Nine Stored Vehicles Report Forms and four Activity Report Forms were also End Shifted during the Pilot Project. Some of the Pilot project Troopers and Supervisors experiences a few issues while using the TraCS Pilot system. However, overall, the project was highly successful and Oklahoma had a working solution that could considerably improve their data timeliness and correctness.

CITS had an exit survey that was given and answered by the Pilot Project Troopers and Supervisors. Some of the positive aspects include the following [23]:

- Troopers found TraCS Crash Report Form easy to use and it saved them significant time.
- Several Troopers said that, if the peripheral devices were functioning correctly, the Crash Report Form could be filled out much faster than the paper version. However, they also pointed out that, if the devices failed, then the TraCS version of the forms would take much longer to be filled than the paper version.
- End Shift and Start Shift procedures were easy and smooth.

- Troopers were enthusiastic with one of the crash diagram tools, Easy Street Draw, saying that it was easy to use and well designed plus it produced professional looking crash diagrams.

Some of the problems reported by the Troopers and administrators include the following:

- They felt TraCS was complicated and confusing.
- Many Troopers lost forms due to the End Shift procedure failing. Indications were given that it had been End Shifted successfully when they actually were not. The same issue was reported from the supervisor side; they would reject a form, but the Trooper would never get it back.
- TraCS did not provide its user with an autosave option, so if the program was prematurely terminated for any reason, any form the Troopers were working on would be lost.
- Many of them reported difficulty using TraCS because of lack of computer resources. Additionally, it could take up to 45 minutes to load TraCS, and switching between applications would be slow.

In conclusion, the TraCS Pre-Pilot and official Pilot were a success, and CITS demonstrated that TraCS could be deployed to OHP to improve traffic records with only minor engineering improvements.

3.2 Crash Reporting System (CRS)

To collect information about CRS, I interviewed one of the heads of the CITS group, Dr. Joseph Havlicek [46], along with current and former developers of

the group [5, 24, 71]. Dr. Havlicek explained that CRS was developed as a web based system that can be deployed for agencies that do not have the IT infrastructure for TraCS. CRS would provide these agencies with similar electronic form submission for Crash Forms so that the problem of data timeliness and data accuracy discussed in section 3.1 can still be solved even for agencies who can not afford the infrastructure that is needed to run TraCS.

As Wolff explained during [71], the project was piloted with strategically located agencies such as, The City Of Sand Springs, Durant, Calere, Hugo and Eufaula Police Agencies.

CRS was a success in serving these agencies and in helping The State of Oklahoma to continue improving its data timeliness and data accuracy [46]. Additionally, as Nguyen [24] and Wolff [71]- some of the main developers of this project- explained, all of these agencies are still actively using CRS to report their Crash Forms.

Due to concerns regarding the cost of keeping CRS validation rules in sync with the validation rules in TraCS, agency support, and the IT infrastructure of a long list of agencies, the CRS was not deployed to more agencies.

These days, CITS still receives requests from other agencies that would like to use CRS. However, as Wolff mentiones in [71] they are not permitted to allow access to any new agencies. It is also important to point out that CRS only supports crash forms and does not support citation ticketing.

During [71], Wolff explained that the current version of CRS is not actively maintained as no funding is assigned to it. Therefore, it operates in the same manner as it did when it was originally developed eight years ago.

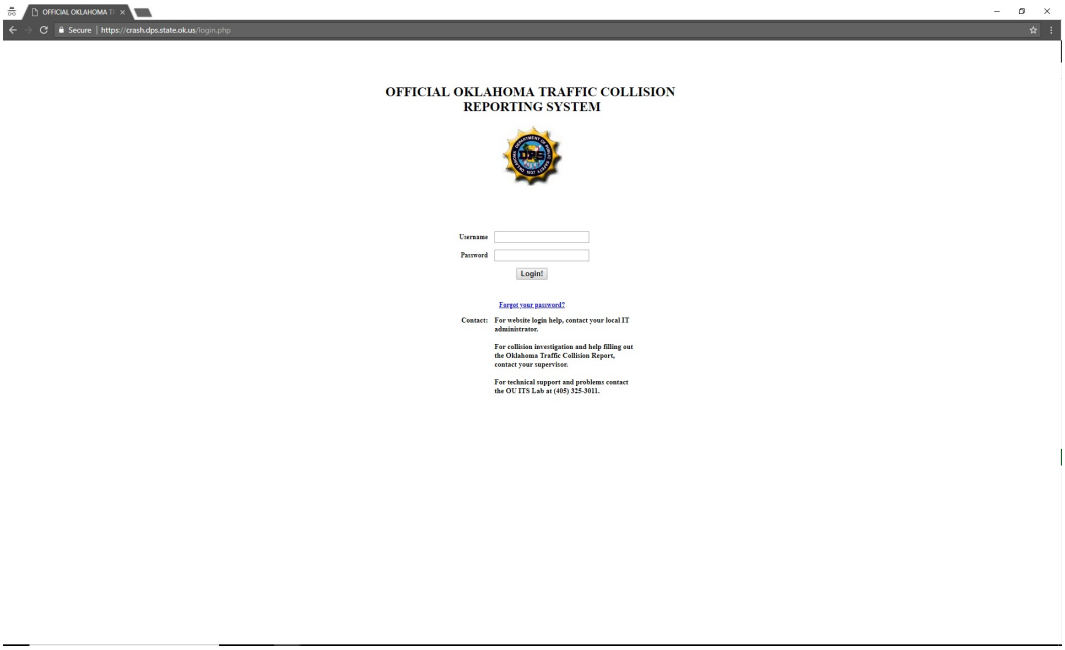


Figure 3.3: Current CRS login page render on an up-to-date version of Google Chrome Browser.

She mentioned that it is still compatible with most browsers, but is not user friendly to operate on small screens like mobile devices. Some of the principal problems with CRS, as Wolff explained, are that the forms have not been updated since it was developed and there are formatting issues that still exist. Additionally, the validation rules are out of date. There are currently no future plans to update CRS and its future availability to the agencies that still use it is currently unknown.

3.3 PARIS, TraCS Successor

According to the contract mentioned previously [47], the completion of TraCS Pilot Project along with the Oklahoma Highway Patrol and the online Crash Reporting System provided to some agencies, demonstrated the potential for

electronic acquisition and management of data to improve DPS Records Management Systems with regard to timeliness and data quality.

TraCS was a successful and risk-free solution, and as The State of Oklahoma had already improved their data timeliness and quality, they were looking to add more features to continue improving their solution. However, TraCS had a proprietary approach which prevented any other state from obtaining the source code and making changes to it [23]. Only one private company was authorized to implement changes to the source code [5,24,32,46]. Additionally, future changes to this policy were not anticipated. So, if a particular state, such as Oklahoma, had a software development team and wanted to make a state modification to TraCS, it was very likely that this change could have been made quickly and cost effectively by Oklahoma. However, as only one company was allowed to modify the source code; it was not possible for states to implement their own changes. As explained by Dr. Havlicek [46] and the CITS report [23], to be able to make changes, each state would have to implement these changes by contracting the company that had the authority to make the changes and the changes would then be made available to all states that are members of the coalition at no cost. The other way in which a state could make changes or enhancements, as the report [23] mentions, is by building a support for the changes in the National Steering Committee. However, as explained in [5, 23, 32, 46] even when a state was able to obtain a contract with the private company to build their changes, there were situations in which the software development resources of the company were fully utilized by the needs of the National Steering Committee. Under those situations, the Iowa DOT or the National Steering Committee had the authority to prohibit individual

states from contracting the company for state specific modifications.

At this point, Oklahoma had decided that their best option was to start building their own solution. Since they had a positive experience on previous projects with CITS group, they asked them to build their new system. The new system they asked for needed to be able to grow, change and scale as much as was necessary.

Based on the contract [47], the main concept of PARIS was to have a system that would transmit electronic forms to a physical or logical subnetwork (DMZ) located at the University of Oklahoma instead of DPS. This secure server would have error checks, validations, and translate the data to a format identical to what TraCS was using. In general, it would simplify and improve the security of the DPS private network.

From the contract mentioned earlier [47], we can see that compared to TraCS, PARIS was intended to have the following qualities :

- Be simpler, more efficient, and more cost effective. Thus, allowing DPS to deal with a single source of external crash data arriving in a unified format from a secure source
- Improve security of the DPS private network
- Improve reliability and performance regarding fixes, network outages, and data formatting problems
- Provide an interface with third party software like MobileCop including NCIC queries
- Have built-in ID scanner support

- Have a crash finder tool that would allow users to fill out all of the crash data information with only a couple of clicks
- Work offline and synchronize data when it is back online without loosing any forms. TraCS assumed the transmission would always work and many times Troopers were loosing their forms as the transmission did not got through and TraCS did not keep a copy of their forms locally

PARIS was developed to improve upon the capabilities and performance of TraCS in order to better satisfy Oklahoma's needs. It was tested by multiple Troopers before it was deployed for the first time to the Oklahoma Highway Patrol and later on to the Oklahoma City Police and Tulsa Police [5,24]. These three agencies are the ones that produce the majority of crash form data that comes into DPS as they handle most of the collisions in the state [5, 8, 24, 32]. These are also the places where most of the accidents occur [8]. This can be seen more clearly in Figure 3.4, a report from the Statewide Analysis for Engineering & Technology (SAFE-T) .

PARIS has a high acceptance rate among the officers that use it [24, 32, 46]. In an interview with an officer from Woods County Sheriff's Office [4] who is actively using PARIS, the officer gave it a rating of 10 out of 10 on usability and ease of use. As officer Honeyman explained, Woods County has 7 officers and all of the officers are provided with Panasonic Toughbook laptops that are designed to survive under extreme conditions. However, only one of these seven officers currently runs PARIS. However, because of his positive feedback, the remainder of the agency is scheduled to receive PARIS in the first quarter of 2018. One of the highest selling points of PARIS for this agency

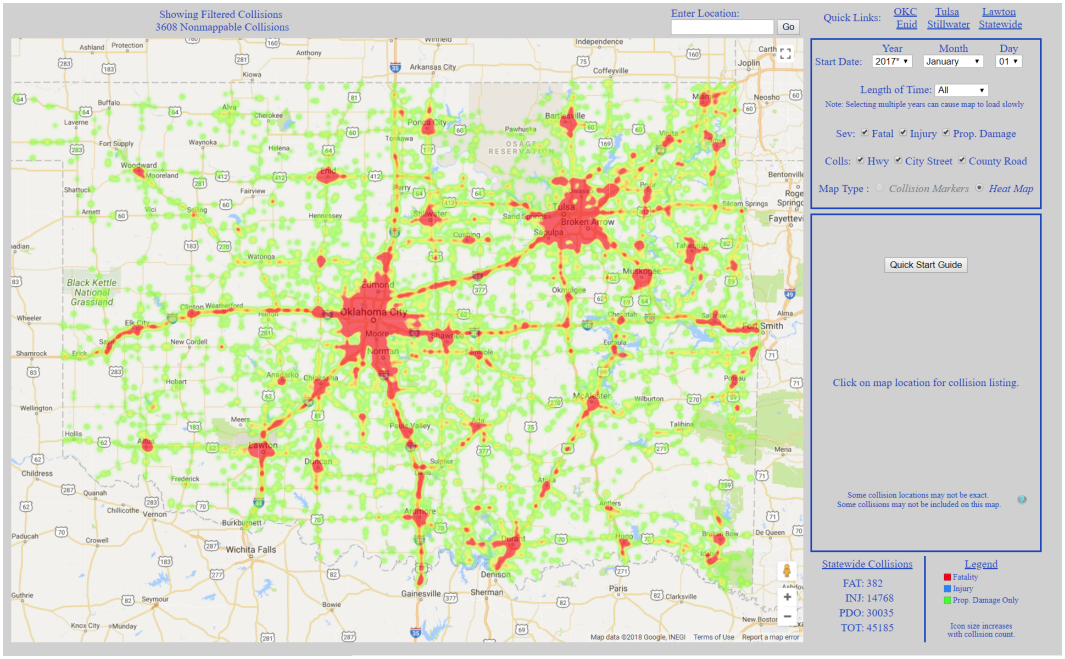


Figure 3.4: Collision report generated using SAFE-T [8].

is that its simplicity greatly reduces the time spent writing citations. This was affirmed by the officer in more than one instance during the meeting. The interviewed officer stated that he is able to fill out a citation in less than a couple of minutes thanks to PARIS' easy-to-use interface and validations. He also mentioned that, even though PARIS has an available offline mode, he has not had to use it since his agency has a reliable internet provider which greatly reduces the chance that internet will not be available.

3.4 Data Flow

As seen in the figure below provided during the interviews [5, 24], there are three digital entry data points in the Oklahoma Collision Data Flow. These points are the CRS, Tulsa Police Department eCitation system, and PARIS.

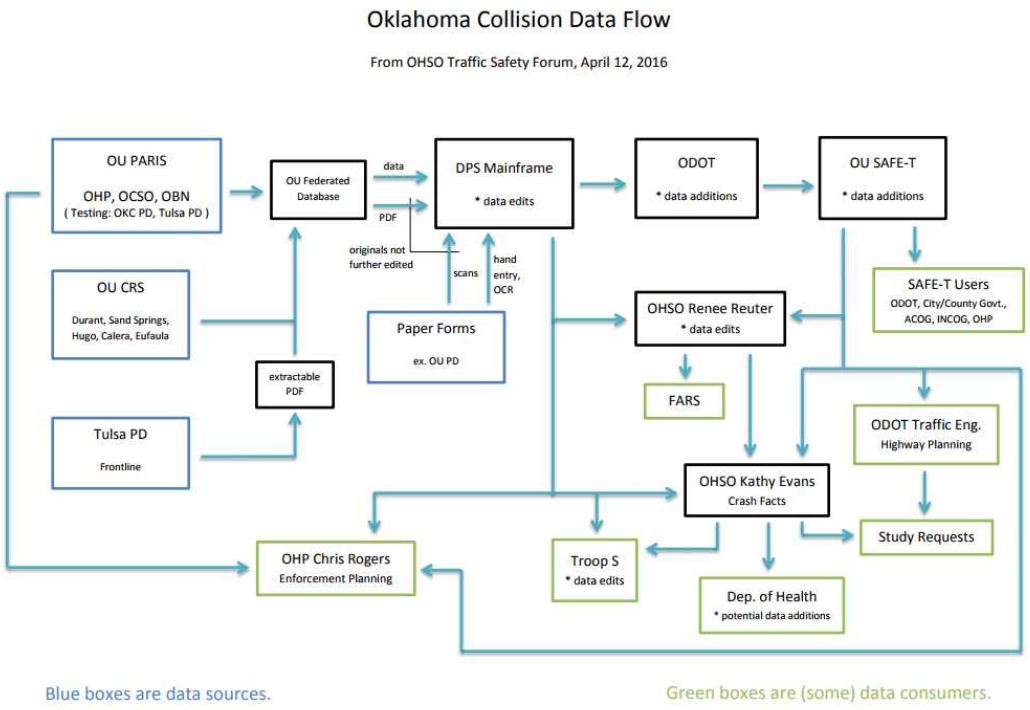


Figure 3.5: Oklahoma collision data flow [5, 24].

The Tulsa Police Department eCitation system requires some adaptation in order to provide the data in the same format as the other two systems. Once all of the data is in the same format, all of the information can then be inserted into the OU Federated Database(also known as Virtual Database- a way to view and query several databases as if they were a single entity). As both Dr. Campbell [5] and Dr. Nguyen [24] explained, the data from the database can then be accessed by the DPS Mainframe, which is in charge of data edits and inserting the paper forms submitted by some police departments like the OU P.D. As seen in the figure **??**, the data can then be passed to the Oklahoma Department of Transportation (ODOT) for quality control. At this time, any additional details can also be added. Once these steps are complete, the data

can be used again in reports using a reporting tool called SAFE-T.

3.5 Problems with the Current Implementations

PARIS was and is a solution to the problems that the State of Oklahoma originally needed to solve. Additionally, most of the officers who have tested it have high praise for it and believe that it makes their work simpler [4,32,46] However, there are many agencies that can not afford to have computers placed in every car or the IT infrastructure and support needed to run PARIS. It is important to mention that PARIS was not designed to solve budgetary problems or to be budget conscious. Although PARIS could not be implemented in every agency because many smaller agencies do not have the necessary amount of computers, most of the officers in these agencies do have smartphones as explained on Section 2.5. These smartphones provide an opportunity to bring electronic citations services to smaller police agencies in Oklahoma and elsewhere.

Other current commercial systems in the market are expensive, and it is very unlikely that they can interface to state databases in the State of Oklahoma, as it is unlikely that they meet state electronic reporting requirements and there are many political challenges [32,47]. Also, none of the other commercial systems seem to consider offline access, a feature from PARIS and TraCS. Although many officers are provided with excellent internet services, having offline access is a crucial feature [46]. In the case that there is an internet issue, having offline availability would ensure that the officer does not experience any problems or frustrations should internet suddenly become unavailable.

Most of the current services do not target smaller agencies. This means that most of these small agencies still have to write citations and crash reports

using paper and pen because they can not afford any of the options on the market [46]. Even if any of the other services offer viable options, it is very likely that the service provider would own the capture data and customization would most likely be expensive.

Most of the current solutions on the market work using expensive printing devices, such as the ones that can be seen on the eCitation Coalition Members with very popular printers manufacturers [30]. Most of these systems work using Thermal Printers but they seem to be only compatible with top quality printers. Acquiring these printers would add extensively to the final budget of the agencies.

Small agencies need a solution that would allow them to digitally collect data in a platform that can run on many devices and is not only less expensive but also simpler to use while still delivering high accuracy data [32, 46]. Therefore, this solution should offer the same features as PARIS while adding the additional offering of a more affordable option.

3.6 Potential Solution

The solution I am proposing will target all agencies that currently do not use electronic submissions because they can not afford the required hardware. This solution is an inexpensive electronic citation system that delivers high accuracy data in a user-friendly manner using existing hardware for agencies such as smartphones and low-price peripherals. As this solution will try to maintain all of the features from the current solution developed by the CITIS group, PARIS, but be a light and cross-platform version. The proposed name for the solution is PARIS+ Mobile. In addition to the solution explained in this thesis,

the CITS group is currently developing the new version of PARIS Desktop, also called PARIS+ as it will come with new features such a redesigned user interface and better data analytics tools.

Since the proposed solution should be inexpensive, a good approach is to use hardware that agencies and officers currently have- such as smartphones. As discussed in the previous chapter, smartphones are very popular and most people are familiar with how to use them [38, 51, 62]. The platform will be an Android and iOS application that can easily and cost-effectively be deployed using the Play Store for Android and App Store for Apple devices (the common places where smartphone users go to download their apps). Since this solution does not have a large budget (initially there was no budget) for development or many developers, the most affordable approach would be to share as much code as possible between the iOS and Android applications. As mentioned in chapter 2, the approaches that allow developers to share code between different platforms are Web, Hybrid, and Cross-platform solutions [64]. Because the platform must be available for offline usage, the web approach must be discarded due to the fact that it needs to reach the server to get the content in order to load. There is a chance that the hybrid approach's performance will not be fast enough to provide the smooth experience that the agencies are seeking. Therefore, the cross-platform is probably the most viable approach to solving this problem.

3.6.1 Choosing a Framework

I analyzed different cross-platform frameworks that allow developers to share the greatest amount of code across all platforms while still having native and

high performing applications. The solutions considered were, React Native [35], Native Script [59], and Xamarin Forms [55].

3.6.2 React Native

React Native is an Open Source initiative from Facebook developed with the purpose of fixing performance issues on their applications. React Native is based on the React web library used to develop web applications [27]. Therefore, its coding and structure are very similar to that of web developing. It is coded using JavaScript, a very popular web language, and allows developers to create both real and native mobile applications [35]. This framework is especially useful in that, according to its documentation, it allows coders to build applications very quickly and reload these applications instantly.

React Native has become quite popular in the development community [36]. In fact, it is no longer maintained by Facebook and instead is currently maintained by the online developer community. The number of developers using it is steadily increasing every month and countless libraries are created using the framework. These libraries are then shared by developers all over the world. As can be seen on the graph from stackoverflow, one of the most popular blogs/forums for developers, most of the questions and activity in 2017 were already coming from React Native.

Although React Native has shows to be quite popular within developers, it still has potential issues. These issues were considered and, based on my experience, I have made the following conclusions. First of all, the libraries are maintained solely by the community. This means that there is a high probability of using libraries that are not maintained because the developers ended up

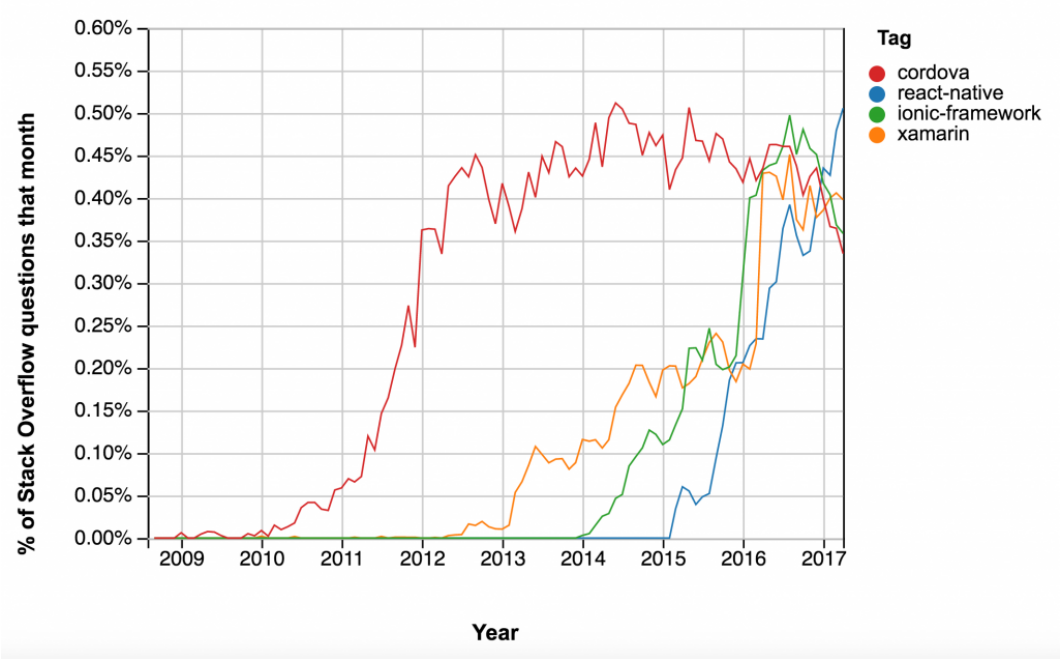


Figure 3.6: Percentage of Stack overflow questions [66].

abandoning the project. This causes major concerns because the project can end up becoming stacked with bugs. These bugs may be more difficult to fix because the issues causing them are usually not related to the solution itself, but to the libraries that it was using. Attempting to solve these types of bug issues can be both time-consuming and expensive. Another potential issue is scalability. Developers must be extremely cautious in keeping documentation up to date when developing enterprise solutions using JavaScript in order to ensure that the product remains easy to maintain. Even though React Native is a viable approach and popular with developers, I have concluded that most of its popularity comes from the fact that building prototypes using it is relatively easy to do. Even though building prototypes using React Native is simple, it is difficult to build scalable applications using it due to the lack of strong types and compilation time error checking of javascript. If we were to

chose this framework, developers would have to be extremely cautious when writing code to ensure that it is scalable and not too costly to maintain.

3.6.3 Native Script

Native Script is another open source framework that can build native applications [59]. Although it is not as popular as React Native, it could still serve as a viable solution. Native Script allows developers to build native applications using Angular (a popular framework used to develop web applications) using either the TypeScript or JavaScript language.

One advantage of Native Script is that the developer can start coding on it using TypeScript out of the box. This is useful because, since TypeScript is a strongly typed language superset of JavaScript, it allows users to have compilation time error checking and makes it simpler to keep maintainable code as opposed to JavaScript [45, 61]. Nonetheless, any JavaScript code is still valid on TypeScript even though is not the best practice to have a lot of JavaScript code on TypeScript.

One disadvantage of Native Script that we saw when making our decisions is that it is not as popular as other frameworks. This means that developing some tools might take a little longer since they will have to be done from scratch.

Native Script works with strongly typed languages and is, therefore, a feasible solution for building a scalable and easy to maintain app. Although Native Script would be a feasible option, CITS has a highly specialized team and has purchased licenses in other frameworks. Therefore, it would be more logical to take advantage of our current expertise instead of investing time and



Figure 3.7: Xamarin forms demo app screenshot from official website.

money into a framework that is completely new to the team.

3.6.4 Xamarin Forms

Xamarin Forms is Microsoft's version of an all open-source cross-platform framework. It allows coders to build native applications for iOS, Android, and Windows while sharing over 96% of the code [55].

Xamarin Forms appears to be a feasible solution for PARIS+ as it allows developers to write mobile applications using C#, a very popular language for desktop app developing. Xamarin Forms allows coders to share not only business logic code but the whole code of the interface [55].

One disadvantage of Xamarin Forms is that it is changing frequently. It is also not as popular as other frameworks, especially because Microsoft's models are not the favorite among the developer community as can be read in forums [10, 65]. One advantage of using Xamarin Forms is that all of the CITS developers are familiar with the language and the group already has purchased licenses to use its Integrated Development Environment (IDE) . This means any PARIS current developer could be able to immediately start developing, improving, and making changes to PARIS+ as necessary without having to invest time or money into a new framework.

After considering the strengths and weakness of the previously mentioned frameworks, the framework selected to develop the application will be Xamarin Forms. Xamarin Forms possesses the same functions as the other frameworks that were considered, plus it will use similar language and IDE than the rest of the projects at CITS, meaning it will be simpler to maintain as any current developer could contribute.

3.7 Printing Hardware

As previously discussed, all of the solutions that were mentioned traditionally use reliable yet expensive printers that increase the overall cost of the application by hundreds of dollars. However, since I am looking for a more affordable solution, other options will have to be considered.

Using one of the more expensive printers on the market would make the process more simple for developers because most of these printers come with Software Development Kits (SDK) for smart-phones that run on iOS. To form a better understanding of Software development kits (a.k.a devkit), think



Figure 3.8: Star thermal printer.

about an SDK like a kit used to build a model car; this kit includes all the necessary pieces and assembly instructions needed to put it all together [7, 48]. SDKs makes it simple and quick to interface with a specific device. Although using cheaper solutions might potentially increase the developing complexity, it is definitely an important option to consider when trying to keep the price of the application as low as possible.

Because of their specific characteristics, the printers used by all of these systems are thermal printers. Thermal printers are a great option for this project because of their solid design; compared to ink printers, they have almost no moving parts. These printers do not need ink replacements and require minimal maintenance. Additionally, thermal printers have been used in envi-

ronments similar to the ones officers face and have demonstrated that they work well under these conditions [41, 81]. These qualities possessed by thermal printers make them the best printer option for this project.

3.7.1 Choosing an Affordable Printer

Thermal printers to be used for this project were searched for around the U.S. by browsing the internet. Although some seemingly reliable and affordable options were found [1, 18, 72], in an attempt to locate even more affordable options, Chinese providers were contacted using the Chinese multinational e-commerce site Alibaba [16]. Through these providers, plenty of options for affordable thermal printers were revealed.

As there are a large number of viable printer options, making the decision became complicated. In order to simplify the search, the requirements were narrowed down to help identify specific characteristics that were considered as necessary for the proposed solution. In order for a printer to be considered, it would have to portable, battery operated, have blue-tooth connectivity, and at least 80mm width. Plenty of emails and calls were exchanged until a seemingly feasible provider was found and the first order was placed. Even though none of these printers come with an SDK, they did come with helpful examples on how to use them to connect with native applications.

The printers finally arrived in the U.S. and were tested. They worked surprisingly well and were faster than originally expected. On top of this, the connectivity process turned out to be quite simple. However, a driver still needs to be developed to be able to use the printers with the application.

3.8 Application Features

Based on the current implementation of PARIS and the other options that exist in the market, there are some features that this solution must have in order to be considered successful. The features that are vital to the success of this project include speed, simplicity and affordability. If our product makes it easier and quicker for officers to fill out their citation forms and is affordable, then it can be considered a success.

3.8.1 Secure Login

It is important that the application be secure enough so that only the owner of the phone can be granted access to it. To make this possible, the backend API that will be developed should only accept authenticated requests for all of its routes- except for the login route. In order to perform a log-in, the users should provide their correct unique username and password. Failed authenticated requests will be handled by the API responding with an error that prevents access to invalid users. The authentication used will be a token-based authentication. A token is a system randomly constructed representing the user or identity it claims to be [82]. A token-based authentication embodies the exchange of client authentication credentials, such as username and password, for a server generated authentication token [34]. The API will be be running under proxies to add an extra layer of security against brute force attacks.

3.8.2 Barcode Scanning

This feature can basically be considered a standard in all eCitation systems as all of them have it. All U.S. driver's licenses have a human-readable front side



Figure 3.9: Arizona-U.S. driver license example.

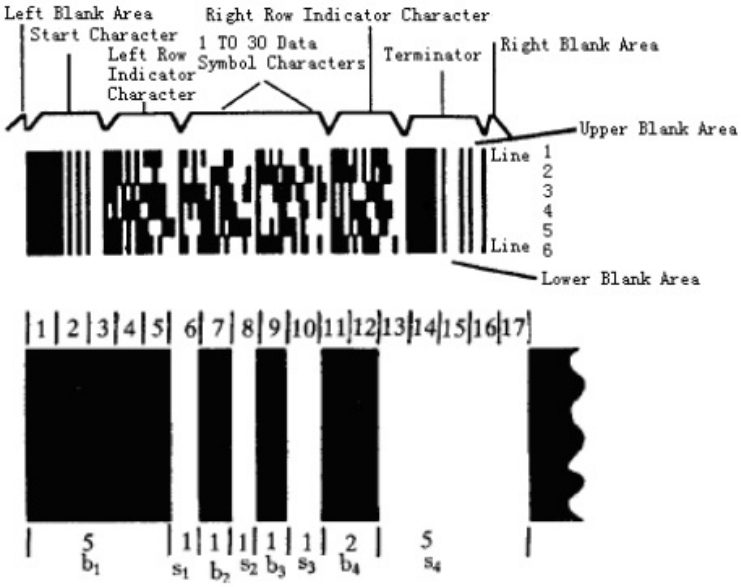


Figure 3.10: Structure of bar code and symbol character [67].

where basic information can be found about the driver, including their picture. On the back side, most U.S. driver's licenses have a barcode that can provide all of this information digitally encoded in the bars [2].

The barcode used on U.S. driver's licenses is called PDF417 [2]. PDF stands for Portable Data File and this barcode is capable of encoding over a kilobyte of data per label [67]. The symbols on the barcode are 17-modules wide and always consist of 4 bars and 4 spaces. See Figure 3.6 [67].

It would be ideal to include a feature that has an external hardware device which scans the barcode in just milliseconds. However, this would increase the hardware price and add yet another device that would need to be maintained. Therefore, the idea of using an external device for scanning barcodes will have to be discarded for now.

Since this solution needs to kept as simple and inexpensive as possible, the proposed approach is to make use of the existing camera on the smartphone devices. The smartphone camera will be used to input the barcode as an image and then have code process and decode the barcode of the ID.

3.8.3 Validation Rules

In order to make the system simple and quick to use, it is important to have strong validation rules as mentioned in a interview by Captain Cannaday [32]. These rules are important in preventing the users from submitting an incomplete form or a form with incorrect data. As it was discussed on Section 2.2 it is necessary to develop a validation data entry and prevent the user from typing as much as possible to minimize errors and provide a good user experience. In order to execute this, it is best to have drop-down menus that only allow the user to select data instead of typing it in. This feature's main purpose would be to prevent typing mistakes. The validations should also be clearly visible so that the user can easily identify any errors in filling out the form. This will

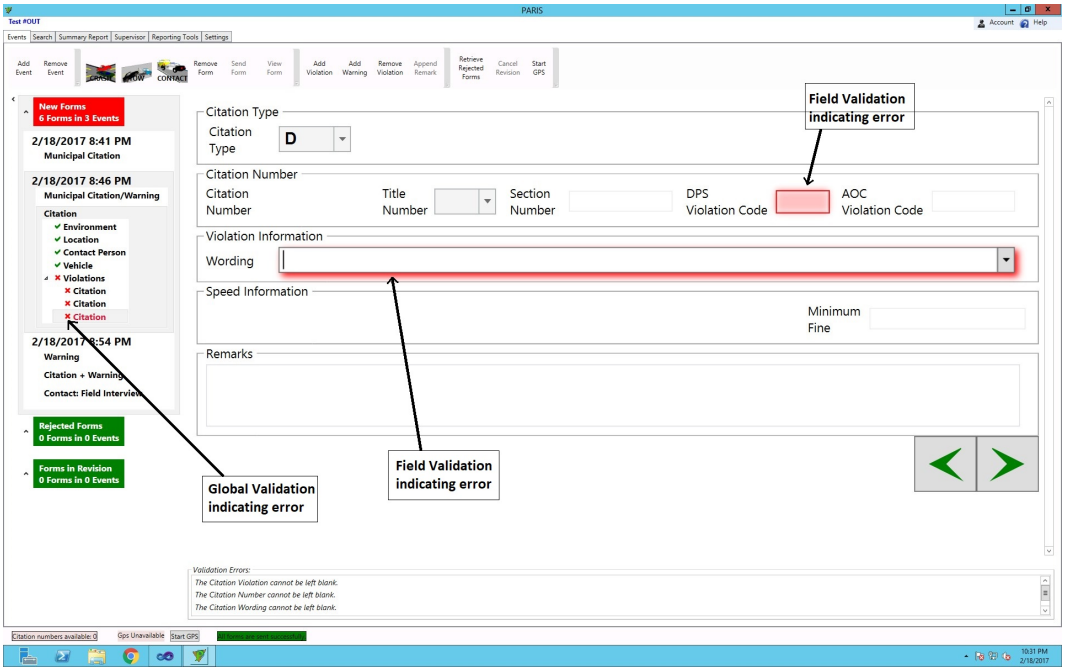


Figure 3.11: PARIS Screenshot showing its validations.

be made possible by including a validation message directly alongside the field and a global indicator for each part of the form. See Figure 3.5 for reference.

3.8.4 Add Multiple Violations

One of the features that makes PARIS so successful is how easy is it for an officer to add additional violations to one driver within just a couple of seconds. See Figure 3.6. Before PARIS, by using the handwritten approach or even some of the current systems, the officer would have to fill the whole form out entirely from the beginning. In these cases, the officers would be repeating a lot of information which would make the process of adding violations difficult and tedious to complete. For instance, before PARIS, if a driver were speeding while not wearing a seatbelt, the officer would usually just write a citation for the highest fault- in this case, speeding. This would mean that the officer would

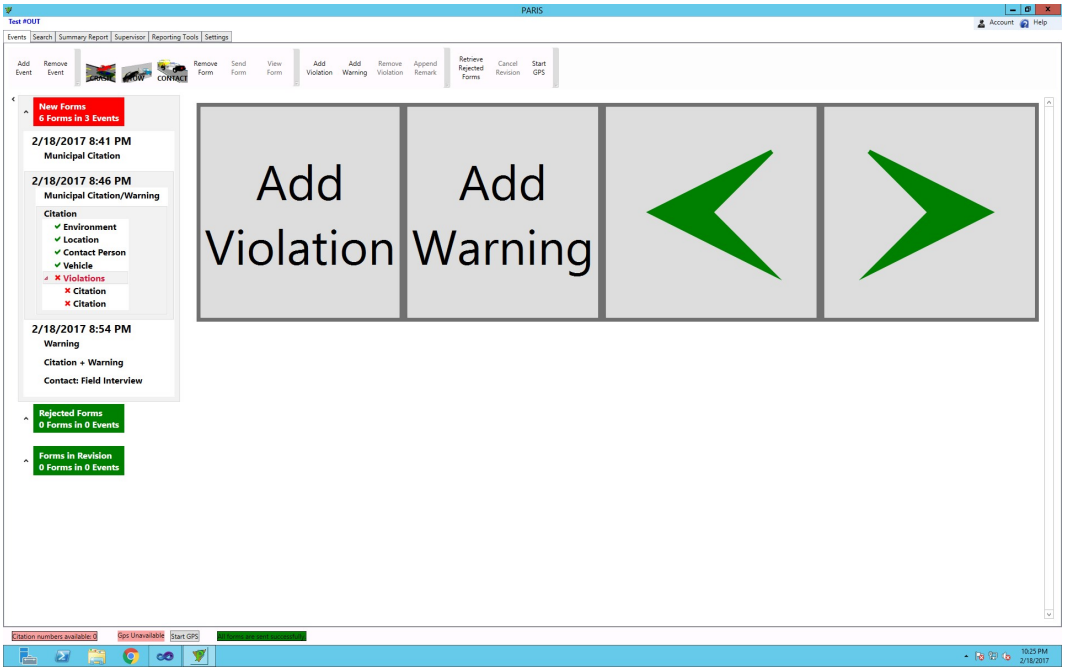


Figure 3.12: PARIS Screenshot showing the menu to add more violations or warnings.

neglect to write a ticket for not wearing a seatbelt. By doing so, the driver would not be punished for his actions and agencies would lose the income from that ticket. This could easily be prevented if the process of adding a violation was made simpler.

3.8.5 Crash Finder

The application should ideally have a location tool that would allow the officer to drag and drop in order to select the exact location of the accident and automatically fill the information required for the citation form. To do that, the app will have an embedded database with all of the streets, highways, private buildings, intersections, cities, and counties with all of their respective GPS coordinates. Having GPS coordinates of all these areas will allow the

application to calculate the closest highway and intersections needed for the form.

3.8.6 Citation Generation

The application needs to be able to render a citation following a provided template. This same rendered citation should be able to be printed so that the officer can give it to the violators. The first approach to develop this part of the application will be to have an html citation rendered on an embedded web browser inside the application.

3.9 Application Back-end

The server for the application could potentially be any server operating system, as it will be running an API developed using NodeJS- a JavaScript runtime built on Chrome's V8 JavaScript engine. This server will be independent and its only purpose will be to communicate with the client application. The API will be inserting data into a MySQL database, following the same structure as the current PARIS system. The API will allow the client application to authenticate users as well as insert new citations to the database. However, the API will not allow the client to edit or delete data. The API will also be in charge of submitting the data inserted by the client to other services.

NodeJS was specifically chosen because it can be run on any server operating system and is easy to develop and prototype. Plenty of libraries can be accessed through its 'Node Package Manager.' This will reduce the developing time significantly because not all of the code has to be written from scratch.

Server-side validations could potentially be a useful feature to include. However, since the database design of PARIS already has validations and the client side will have them as well, it is not crucial to include this particular feature in this potential solution and will, therefore, be left out.

This private API could also later be made open to other clients so that any future development could interface with this same server and allow data to be submitted through it.

3.10 Citation Manager

To make this solution even more user-friendly and unique, a web platform for managing citations will be added. This management tool would allow any court that would like to enroll in the system to be able to have the citations submitted by the system instantly and be able to print and review them. This tool is especially important in Oklahoma because the court's current efforts to provide all of the requirements to allow more agencies to submit data digitally are not enough. These methods usually end up taking too much time to add to their system.

3.10.1 Client Side

This tool will have the capabilities to query data from a desired period of time and will allow the users to pass different filters like cities, names, and officers, to make searching a simple and smooth experience. One of the more challenging aspects of this is that this tool should be able to manage an extremely high volume of data in order to exceed the high expectations of the court.

To make this app both quick and simple to use, it will be necessary

to make it a web app using a single page design. A single page web app is a website that only reloads the content that needs to be updated, leaving the rest of the website as it is and maintaining the state of all components that have not been re-uploaded. One of the popular frameworks for developing this kind of applications is Angular- this will be the framework used in this part of the solution. Angular will be used because its design patterns make the developing process both scalable and easy to maintain. This application will be communicating with a different server than the one the mobile application is communicating with.

3.10.2 Server Side Web API

This server will be only storing citation data and pdf versions of the reports submitted to allow the court members to access any citation at any time. The mobile application server will be in charge of passing the information that the mobile application is submitting to the citation manager server. The server-side API (Application Programming Interface) will be developed using NodeJS and will allow authenticated requests to edit and read citations for the client of the citation manager. It will allow requests to insert data from the server of the application.

This API is restricted to be used only by the client side of the citation manager and the client application's server. However, it is capable of handling much more and could potentially be another service that any other client such as PARIS (or any other private companies' client) could communicate with in order to electronically submit its citations.

Chapter 4

Implementation Details

A great deal of contemplation went into searching for the proper solution to the problems currently being faced concerning eCitation applications. In this chapter, I will discuss the implementations of the features mentioned in Chapter 3. To do this, I will be detailing the problems that were faced during this project and explaining the solutions that were used in order to finally achieve the desired goal.

As mentioned in the previous chapter, the framework that was chosen for this application is Xamarin Forms- a framework that allows developers to code applications using the C# language and share 96% of that code between platforms [55].

During the writing process of this application, the collision form in Oklahoma was undergoing a complete redesign. Therefore, any efforts put forth in writing a collision tool would end up being futile. This is because once Oklahoma releases the new form, all of the front-end and back-end code will most likely have to be completely modified.

As explained in the introduction, this implementation is a shared work between Hesham Makhlouf and me. My personal contributions on this project are the following:

- Development of the Application Interface(API) for PARIS+

- Structure Organization of the project libraries
- Login Interface
- Printer Driver
- Citation Page Holder
- Bar Code Scanner Interface
- MVVM Library
- Legacy Dropdown Library
- User Control Library
- PARIS+ Ciration Manager Client and Login
- PARIS+ Citation Manager Search Tool
- Development of the Application Interface(API) for PARIS+ Citation Manager Tool

These parts of the application were designed equally by Makhlouf and me:

- Interface and logic designed of Environment Page
- Interface and logic designed of Person Page
- Interface and logic designed of Vehicle Page

```
public string GetPlatform()
{
  var path = "unknown";
  #if WINDOWS_PHONE
  path = "windowsphone";
  #else
  #if __SILVERLIGHT__
  path = "silverlight";
  #else
  #if __ANDROID__
  path = "android";
  #else
  #if __IOS__
  path = "iOS";
  #else
  #if __TVOS__
  path = "tv";
  #else
  #if __WATCHOS__
  path = "watch";
  #endif
  return path;
}
```

Figure 4.1: Shared class example.

4.1 Xamarin Forms Overview

As previously mentioned, Xamarin Forms is the framework that has been chosen to be implemented for this project. This portion of the chapter will serve as an introduction covering the important factors that lead to making this decision. Xamarin Forms is a mobile app development framework with UI (User interface) technology that allows users to share not only the logic code but also the complete UIs.

Xamarin allows users to utilize two different code strategies- Shared Project and Portable Class Library (PCL). The main difference between these two strategies is that Shared Projects allows for the use of compiler directives in the code which allow for a range of platform-specific operations. This performs

```
public interface IBluetoothPrint
    {
        void connect();
        bool print(string text);
        bool println(string text);
        bool printImage(string fileName, int width = 48);
        bool print(byte[] cmd);
        void printWebView(CustomWebView webview);
    }
```

Figure 4.2: Bluetooth interface.

perfectly well for prototypes or small projects. However, it does have the potential to become increasingly complicated as projects start growing. See Figure 4.1.

PCL, on the other hand, allows for cleaner implementations as it allows for the separation of platform-specific code into other libraries or files using Dependency Services. PCL has been recently deprecated and the .NETpowe Standard Class has become its successor. Migrating PCL to the .NET Standard is ultimately not too complicated and performing this migration should be considered in the future.

4.2 Portable Class Library

Portable Class Libraries, as mentioned previously, have been exchanged with .NET Standart libraries. Despite this change, the purpose of the libraries remains the same. PCL holds the majority of the code and is where the shared code is written. All of the logic and the views are written in these libraries and are later compiled into the native applications. Figure 4.3 shows the architecture of an application using a portable class library to share its code. Additionally, it illustrates how all of the data access, business, data layer, and

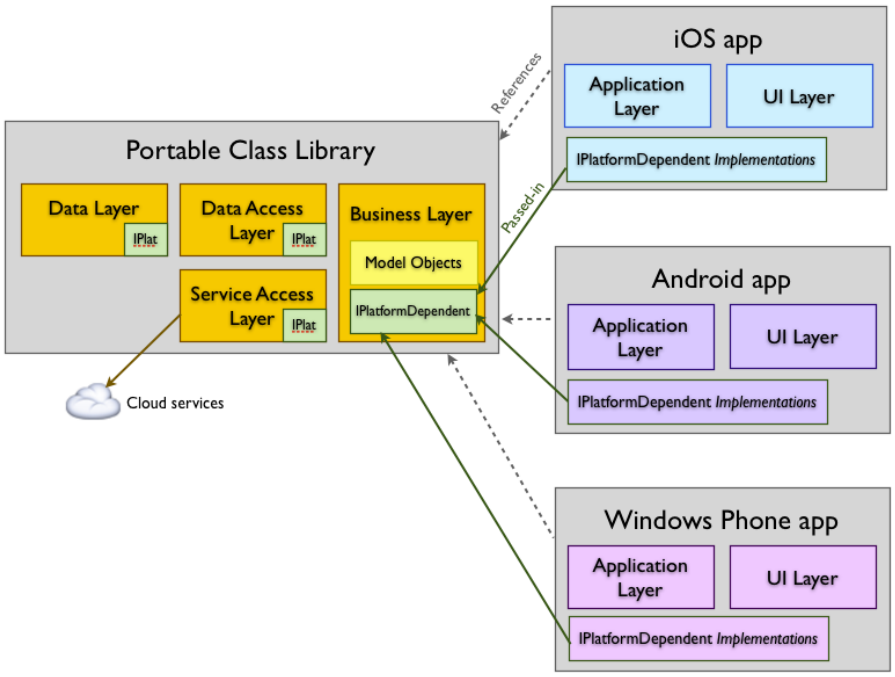


Figure 4.3: Architecture of Xamarin Cross-Platform using PCL [20].

services happen on the PCL while also showing how native features are accessed.

4.2.1 Dependency Services

Dependency Services allow the application to call platform-specific functionality from a shared code. This way, Xamarin Forms can perform any operation that any native application could do. As shown in the diagram on Figure 4.4, the Dependency Service is a resolver- an interface that is defined by the developers. The Dependency Service finds the right implementation of the interface for each platform [17].

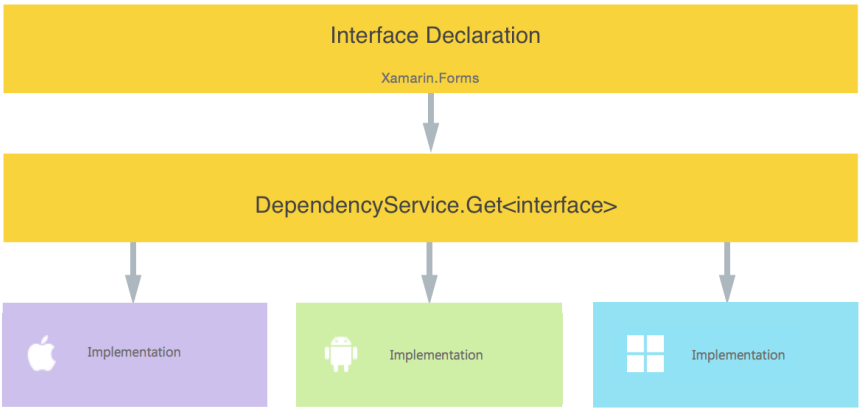


Figure 4.4: Dependency Services overview.

Interface

The interface defines how the interaction with the platform-specific functionality will happen. In simpler terms, the interface declares the functions that are going to be implemented, what the inputs will be for the function, and its outputs. Figure 4.2 is an example of an interface implementation, which is the one used for the bluetooth connection in the application [17].

Platform Implementations and Registrations

Once interfaces are defined, they need to be implemented in the project of each platform that is being targeted. Figure 4.5 shows part of the Android implementation of the bluetooth interface for the application. Each implementation needs to be registered. To do so, the first line of Figure 4.5 is needed.

From this, it can be inferred that the simplest application developed on Xamarin Forms for Android and iOS will have to include at least 3 different libraries- one for the PCL, one for Android and an additional one for iOS.

```
[assembly: Dependency(typeof(Printer.Droid.BluetoothPrintImplementation))]
namespace ITS.Library.Printer.Droid
{
    class BluetoothPrintImplementation : IBluetoothPrint
    {
        public BluetoothPrintImplementation() { }
        public bool printImage(string imagefileName, int width = 48)
        {
            imagefileName = imagefileName.Replace(".jpg", "").Replace(".png", "");
            int id = (int)typeof(Resource.Drawable).GetField(imagefileName).GetValue(null);
            Bitmap image = BitmapFactory.DecodeResource(Forms.Context.Resources, id);
            return print(ImageHandler.getImage(image, width));
        }
        public bool print(string text)
        {
            byte[] buffer = Encoding.ASCII.GetBytes(text);
            return print(buffer);
        }

        public bool println(string text)
        {
            text = text + System.Environment.NewLine;
            byte[] buffer = Encoding.ASCII.GetBytes(text);
            return print(buffer);
        }
        ...
```

Figure 4.5: Bluetooth interface implementation for Android.

Figure 4.6 shows an example of a whole project (solution) for both Android and iOS.

4.3 Project Structure and Libraries Overview

The project was broken up into different libraries (Figure 4.7) to make its maintenance more simple and cost-effective. To try to maintain the principle of Separation of Concern (SoC), each library only contains code specifically for what it was written for (a separate concern) and does not include any additional code.

4.3.1 Model-View-ViewModel Library (MVVM)

This library facilitates the use of MVVM architectural pattern. Using MVVM allows for the separate software structure of views, models, and view models

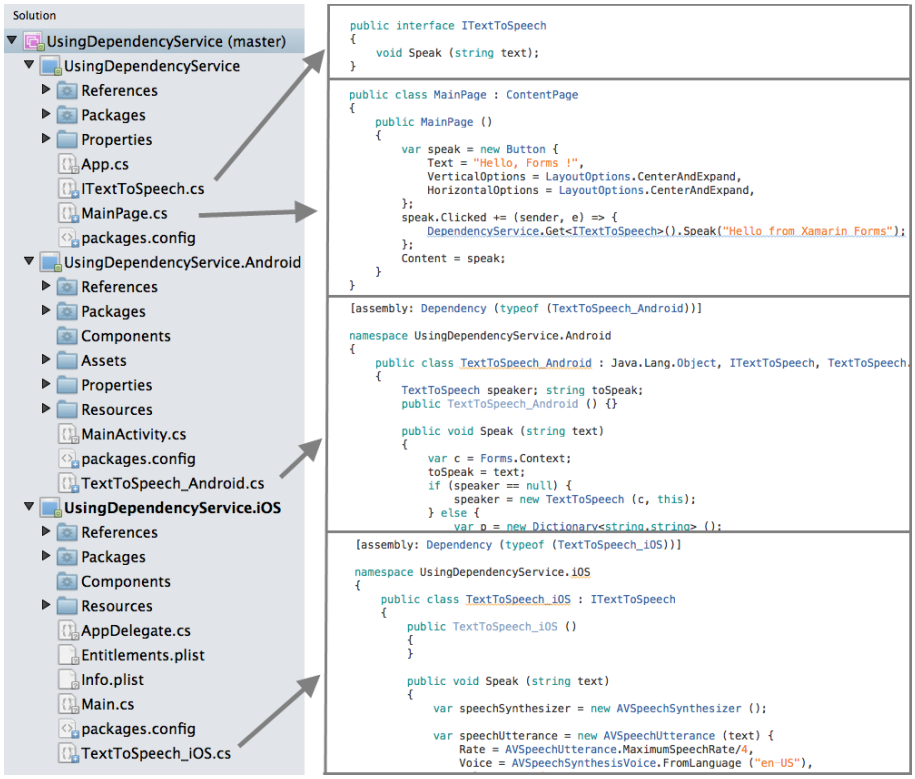


Figure 4.6: Example solution (.sln) for a basic iOS and Android application.

by using the concept of data binding (the connection between user interfaces and business logic).

This means that by using MVVM, pages and logic can be separated to increase the re-usability of the code while keeping it easy to develop, test, and maintain. To connect the pages and the logic, XAML (which stands for Extensible Application Markup Language and is the language behind the visual representation of an application) and Data Binding technology is used [52]. In summary, the views in MVVM communicate with the models through commands and data bindings. The view model can then read the data from the model and display them in the view as seen in Figure 4.8. In [52], XiaoLong Li explains the low level details of how this all works.

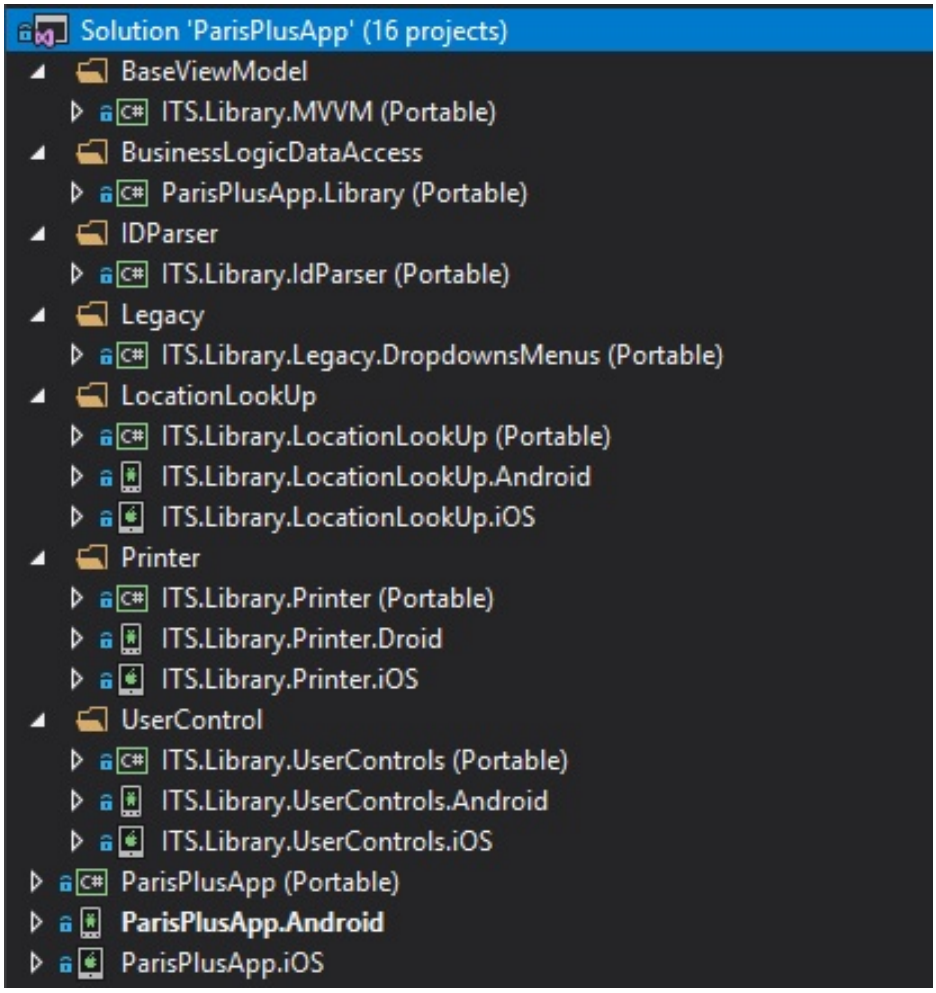


Figure 4.7: PARIS+ libraries overview.

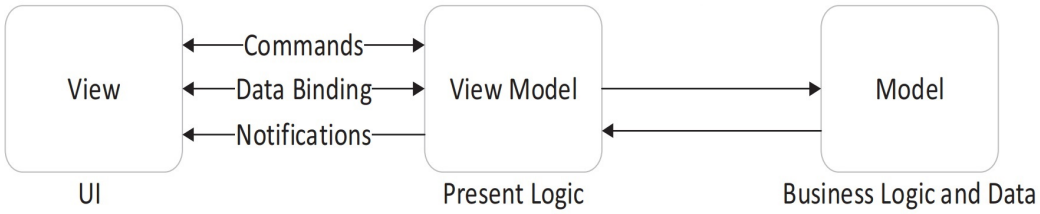


Figure 4.8: MVVM class diagram [52].

This library contains methods that can be used throughout the entire application in order to make MVVM usage simpler. These methods can also

be used as data converters for the views and services to navigate between pages in the application.

4.3.2 PARIS Plus App Library

This library contains all of the model's definitions needed to work on the application as well as all of its view models that allow it to display the data on the views. This library also contains all of the functions needed to make the HTTP requests to the API (application programming interface) in order to save the data to the database.

4.3.3 ID Parser

This library contains all of the code related to the data stored in the driver's license barcode. The details regarding the implementation of this library will be explained later on in this chapter when the features of the application are explained. To use this library, it is necessary to create a new instance of the main class called DriverLicense and pass the text value of the scanner result. Once this is done, the class will parse the fields in the demand calling functions, such as getFirstName(), to retrieve the driver's first name. See figure 4.9 for an example of usage.

4.3.4 Legacy Dropdowns

This library is partially taken from the PARIS Desktop application, but it has been altered and improved in order to work in Xamarin. The library dynamically generates the content for the dropdowns from an xml database file with all of the dropdown values and database keys.

```
DriverLicense dl = new DriverLicense(scanResult.Text);

contactPerson.FirstName = dl.getFirstName();
contactPerson.LastName = dl.getLastName();
contactPerson.MiddleName = dl.getMiddleName();
contactPerson.DateOfBirth = dl.getDOB();
contactPerson.Height = (int)dl.getHeight();
contactPerson.Weight = (int)dl.getWeight();
contactPerson.AddressStreetName = dl.getAddress();
contactPerson.AddressCity = dl.getCity();
contactPerson.AddressZIPCode = dl.getZipCode();
contactPerson.DriversLicenseNumber = dl.getDriverLicenseNumber();
contactPerson.DriversLicenseExpiration = dl.getLicenseExpirationDate();

AddressStateDropDown.AssignSelectedItemFromValue(dl.getState());
DriverLicenceinfostateDropDown.AssignSelectedItemFromValue(dl.getState());
```

Figure 4.9: Driver's license class usage example.

```
CountyNameDropDown = new DropdownViewModel(
    DropdownCache.Instance["citation"]
    .GetDropdown("OklahomaCounties"),
    (string value)=> { contactLocation.County = value; filterCity(); });
```

```
<ITS:ValidatedPicker  ErrorMessage="{Binding contactLocation.Validator[5].ErrorMessage}"
                      IsValid="{Binding contactLocation.Validator[5].IsValid}"
                      Title="Select..."
                      SelectedItem="{Binding CountyNameDropDown.SelectedItem, Mode=TwoWay}"
                      ItemsSource="{Binding CountyNameDropDown.DropdownItems}"/>
```

Figure 4.10: Dropdown usage example.

To populate a dropdown, as seen in figure 4.10, a new instance of the DropdownViewModel(a class from the library) object is needed. For that, constructor(a method that allow to create a new instance of the class) receives two parameters. The first parameter is a dropdown Item which is created by using the DropdownCache static class and passing the type of the dropdown instance (this could be citation or collision) and the dropdown name- OklahomaCounties in this example (The list of complete dropdowns can be found in an xml file included in the library). The second parameter of the class is an action which is executed when the user selects a dropdown.

```
LocationLookup lookup = new LocationLookup(position.Latitude, position.Longitude);
```

```
CountyNameDropDown.AssignSelectedItemFromValue(LocationLookup.co_name);
contactLocation.Latitude = LocationLookup.Latitude;
contactLocation.Longitude = LocationLookup.Longitude;
if (contactLocation.City == "")
{
    contactLocation.NearCity = true;
}
contactLocation.IncidentLocation = LocationLookup.street_name;
ViolationOccuredinTroopDropDown.
    AssignSelectedItemFromValue(LocationLookup.troop_div_name);
CityNameDropDown.AssignSelectedItemFromValue(LocationLookup.cityName.ToUpper());
```

Figure 4.11: Location look-up library example code.

To display the values of the dropdown, a custom control (explained later on in this chapter) needs to be used in the XAML view and it needs the property ItemSource to be binded to the DropdownItems property of the DropdownViewModel instantiated previously. See Figure 4.10.

4.3.5 Location LookUp

This library contains a database with all of the cities, counties, streets and buildings that allow the application to find the information needed for the forms when the user selects a location. This library is explained in more detail later on in this chapter. This library contains three projects, one for the Android implementation, another for the iOS implementation, and a portable class library which allows the shared code to interface with this custom implementations on each platform.

The library's usage is relatively straightforward. A new instance of the LocationLookup Class is required and the constructor simply takes two

```
public void Connect()
{
    //DependencyService.Get<IBluetoothPrint>().connect();
    PrinterHelper.connect();
    _pageService.DisplayAlert("Printer connected", "Connection Succed", "ok");
}
public void Print()
{
    _pageService.DisplayAlert("Printing", "Your citation is being printed", "ok");
    //This will print the current CustomWebView on the Printer Page
    DependencyService.Get<IBluetoothPrint>().printWebView(webview);


}
```

Figure 4.12: Printing library example.

parameters, the latitude and longitude of the user. With that information, it will calculate the information needed for the forms. The results of the class are then stored on properties inside the class. See Figure 4.11 for more details.

4.3.6 Printer Library

The printer library handles the bluetooth connections to the printing device and allows the user to print the citations. This library also includes three projects for the custom implementation on each platform and one that is used in the shared code.

To use the printer library, it is required to first connect to the printer and then call the print method, Figure 4.12 and pass the rendered view that contains the rendered citation as a jpg. The printer library then converts this jpg into a black and white images to finally send the image as Bite array using a socket connection, so that the printer can produce the citation.

```
<ITS:ValidatedEntry  ErrorMessage="{Binding contactLocation.Validator[5].ErrorMessage}"
                     IsValid="{Binding contactLocation.Validator[5].IsValid}"
                     Placeholder="County"
                     Text="{Binding contactLocation.County, Mode=TwoWay}" />

<ITS:ValidatedPicker  ErrorMessage="{Binding contactLocation.Validator[5].ErrorMessage}"
                      IsValid="{Binding contactLocation.Validator[5].IsValid}"
                      Title="Select..."
                      SelectedItem="{Binding CountyNameDropDown.SelectedItem, Mode=TwoWay}"
                      ItemsSource="{Binding CountyNameDropDown.DropdownItems}"/>
```

Figure 4.13: Custom user control usage.

4.3.7 User Control Library

This library contains the custom user controls designed for the application to meet all of the requirements defined in the previous chapter. The custom controls are called Validated Entry, Validated Picker, and Validated Date Picker. The usage for the three of them is similar.

The controls expose common properties, ErrorMessage and IsValid. ErrorMessage is the value that displays to the user when the input has an error. IsValid represents the current status of the control that could be true or false, meaning it is valid or not. The custom entry has a binded property text that holds the value entered by the user. The picker has bindable ItemsSource which are the values in the list to pick, and the bindable SelectedItem property that holds the value selected by the user.

4.4 User Interface Overview

The interface was designed with the intention of making the program as quick and simple as possible to use. Directly typed input fields were avoided wherever possible as officers typing using their smartphone keyboards take extra time and may lead to unnecessary mistakes in fields that are difficult to validate. In

order to provide a solution to this particular issue, dropdown or picker menus were used as much as possible since the simplest way to pick a value from a touchscreen is to click on it. However, since some customization was needed, the traditional drop-downs provided by the frameworks were not enough. To resolve this issue, custom implementation (such as text entries and pickers) was necessary. To make the experience simpler and smoother, the citation form was divided into different sections similar to the PARIS Desktop Application mentioned in previous chapters. The parts of the form are divided into the following sections:

- Location Lookup or Crash Finder

  Here the user can select the exact location where the contact occurred from a map. This way, the application is able to process the rest of the necessary information related to location and fill it automatically.

- Environment Page

  This section holds information about the date and time of the incident, the type of incident and the court date if needed.

- Location Page

  This section of the form displays information regarding the location of the contact. The majority of this section can be automatically populated by the information provided in the Location Lookup.

- Person Page

  This is where the officer provides information about the driver. Some of these fields include name, nationality, gender, and race. Most of this

data can be automatically populated when the user takes advantage of the barcode scanning feature.

- Vehicle Page

  This section contains information about the car that was being driven during the incident. This information includes the license plate number, color, and make of the car.

- Warning and Violation Section

  In this part of the form, the officer is able to add any additional violations or warnings that will be given to the driver.

- Preview and Printing Page

  This page shows a preview of the citation, lets the user zoom in to double check fields if needed and allows the user to connect to the printer and print out the citation to give to the driver.

4.5 Authentication

Both the API and the Client Application (referred to from now on as PARIS+) will have authentication. The authentication on PARIS+ prevents unauthorized users from passing the login view and the API authentication will block any request responses unless they come from an authenticated user. This login page allows the user to type in their username and password and submit the data to the API, thus making the POST request necessary for the API to perform authentication.

The API then processes the credentials received and makes sure the username actually exists in its database; if it does not, the API will finish the authentication by sending a 401 error reply to the client denoting unauthorized access. If the username entered exists, the API will proceed to hash the received password and compare it with the one stored for that user in the database. Only if the hash password matches the one in the database and the account of the officer is active, the API will reply with a token (JWT explained as planned on the previous chapter). This token will be used on future requests to identify the user and on the time-to-live (TTL) as seen in Figure 4.14, which gives an overview of the process to authenticate the user. This TTL can be adjusted to any requirement needed but, for our current purposes, a token can last for up to 14 days. The complete process to verify the user request to authenticated the user before a JWT is generated as planned in the previous chapter can be seen in detail on Figure 4.15.

Passwords on the API are stored hashed. The hashing algorithm employed for this project is bcrypt. Bcrypt was chosen due to the fact that it is one of the most secure and widely used hashing functions. The security provided by bcrypt comes from the fact that it uses a salt to hash each user's password instead of a global salt for all of the users (using a global salt for all users makes it simpler to hackers to gain access to an account by using brute force). Additionally, the function repeats the salting in order to increase computing time a known number of times. In other words, as the password will be hashed one time using the salt for the user and then it will be hashed again using the same salt and again and again until the predefined number of times the password needs to be hashed is reached. This makes it more difficult

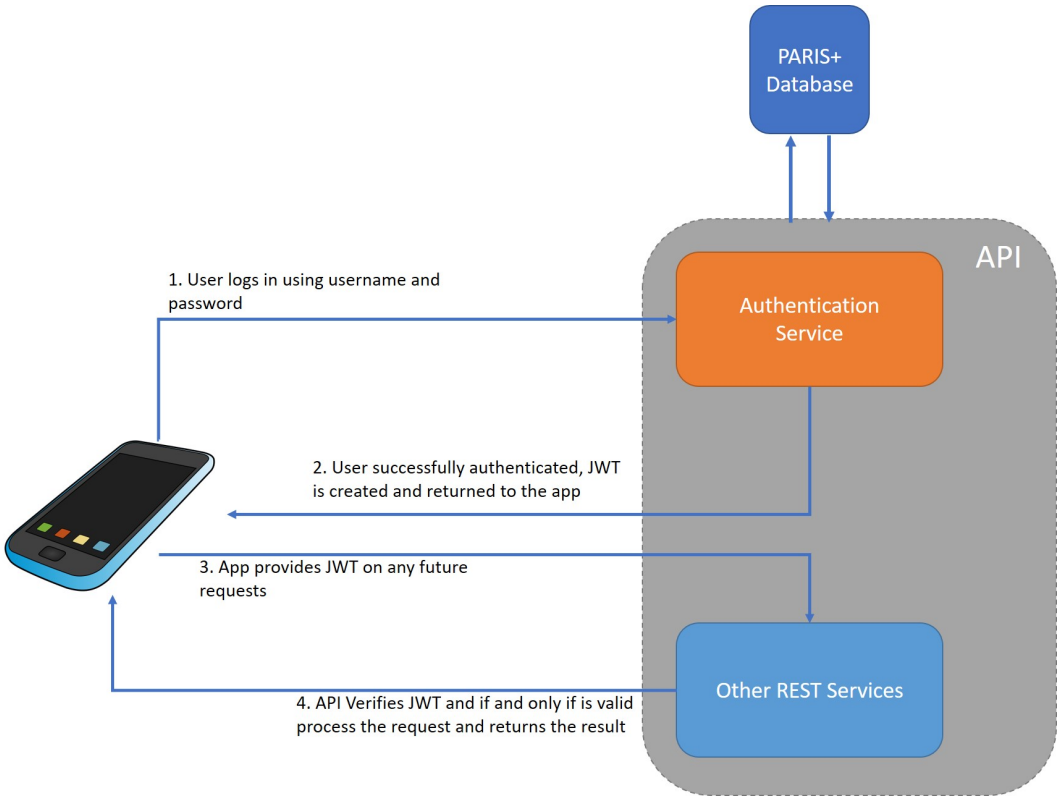


Figure 4.14: Authentication block diagram.

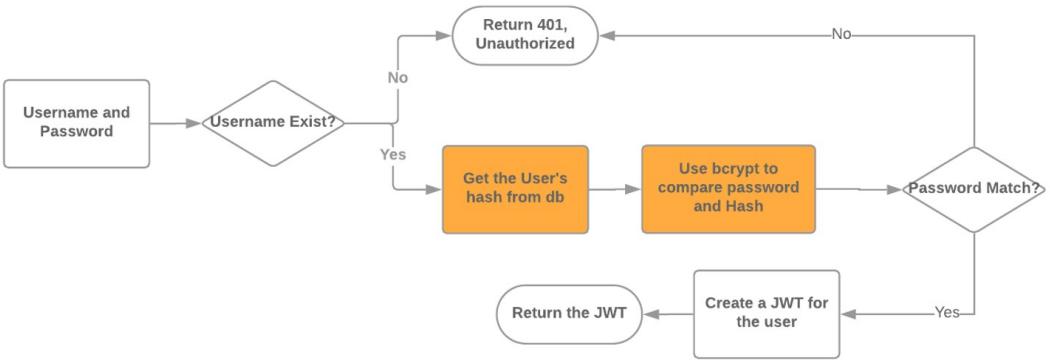


Figure 4.15: Authentication flow chart.

for hackers to attack by using brute-force as they will have to go through a lot of computations for each user they try to hack. Accounts are managed by the administrator by manually making requests to the API authenticated

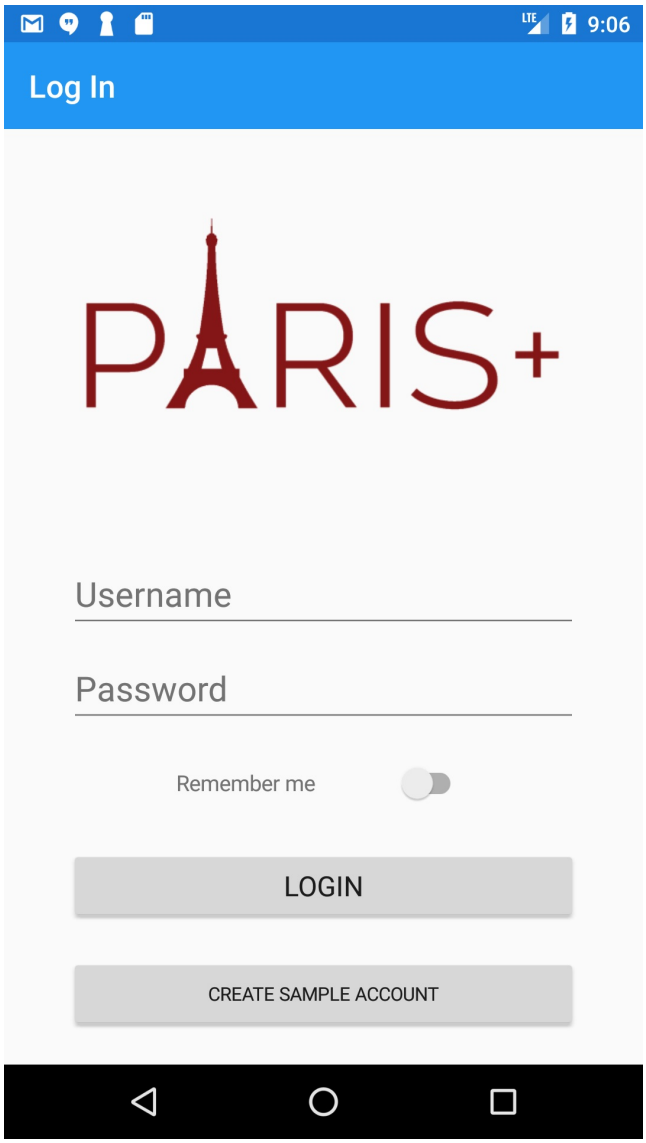


Figure 4.16: PARIS+ login page.

as an administrator. So, no user can recover passwords, create accounts, or change any information about their accounts; only the administrator is set up to perform these tasks.

The login interface, Figure 4.16, has two text box inputs that allow the user to type their credentials along with a submit button at the bottom of the

```
buttonScan.Click += (sender, e) => {

    var scanner = new ZXing.Mobile.MobileBarcodeScanner();
    var result = await scanner.Scan();

    if (result != null)
        Console.WriteLine("Scanned Barcode: " + result.Text);
};
```

Figure 4.17: Barcode calling example.

page. It is designed to be simple and to feel familiar to any other application's login. This way, the users will be accustomed to working with these types of interfaces, thus reducing the need for training.

4.6 Barcode Scanning

The barcode scanning feature is quite useful since it allows the officers to import the driver's data quickly and easily. Another reason for choosing Xamarin is that it comes with a Components Library where coders upload useful libraries that can be either free or paid. These libraries are extremely beneficial because they eliminate the need to create code from scratch.

For the barcode scanning feature, the ZXing (Zebra Crossing) library was chosen. It was decided that it was best to build the scanner using a library because it reduces both the cost and time of development. Using the library allowed for the addition of scanning capabilities without sacrificing any extra developing time. The basic way of using the feature is to trigger the scanner by user input, such as a click of a button, and then call the scanner for a scan using an asynchronous call (Figure 4.17). The next step, as explained in the flowchart on Figure 4.19, would be to verify that the value of the returned output is not null and then pass it through the parser code to fill the form.

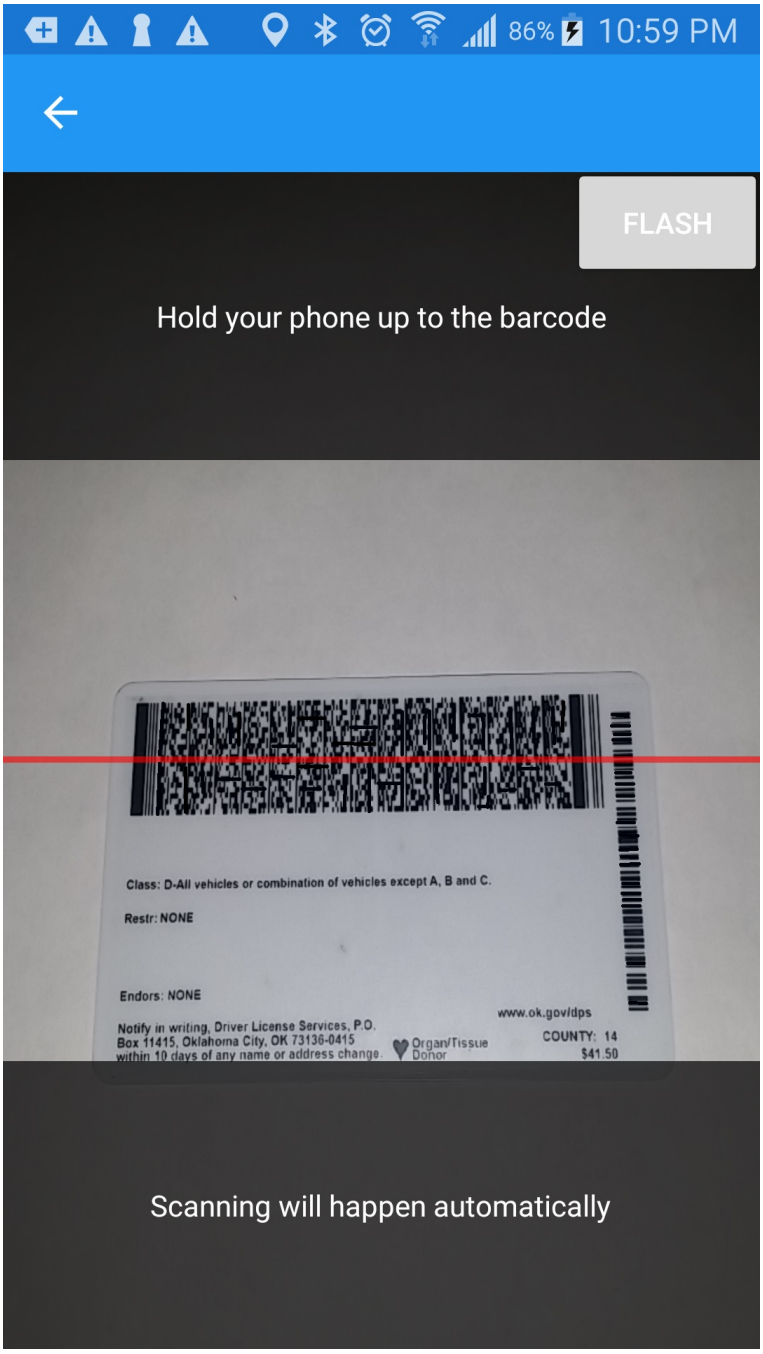


Figure 4.18: Barcode scanning on the application.

The interface for the barcode scanner is designed to be straightforward; it simply displays the camera input with a line to help align the driver's license,

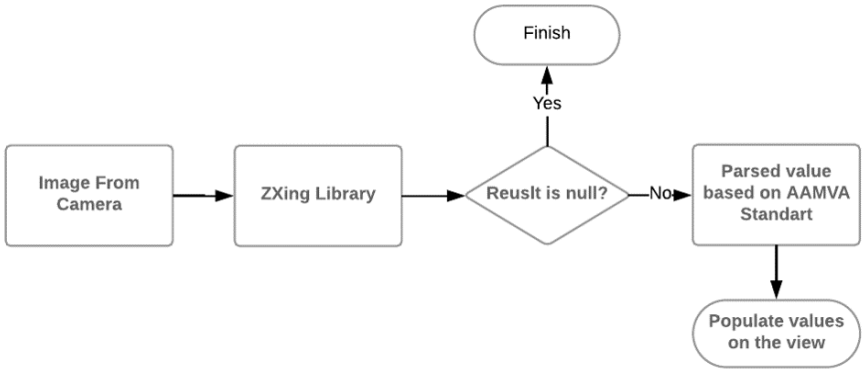


Figure 4.19: Barcode scanner flow chart.

a button to switch the phone light on or off, and provides audible feedback when the user successfully scans a driver license. See Figure 4.18.

The result delivered by the scanner is a text input representing the information encoded in the barcode. This data follows the American Association of Motor Vehicle Administrators (AAMVA) standards [2]. The parser for the standard was developer by another member of the CITS team.

4.7 Validation Rules

In order to ensure the quality of the data, it is extremely important to verify that the data inserted by the user is accurate. To do this, two different types of graphical error indicators are required on the interface.

To accomplish this, a validator library was developed (User Control Library mentioned previously) that receives parameters such as the total number of validations on that view of the form, the status of the field, and the error message to display on the interface. Developing this validator was a challenge

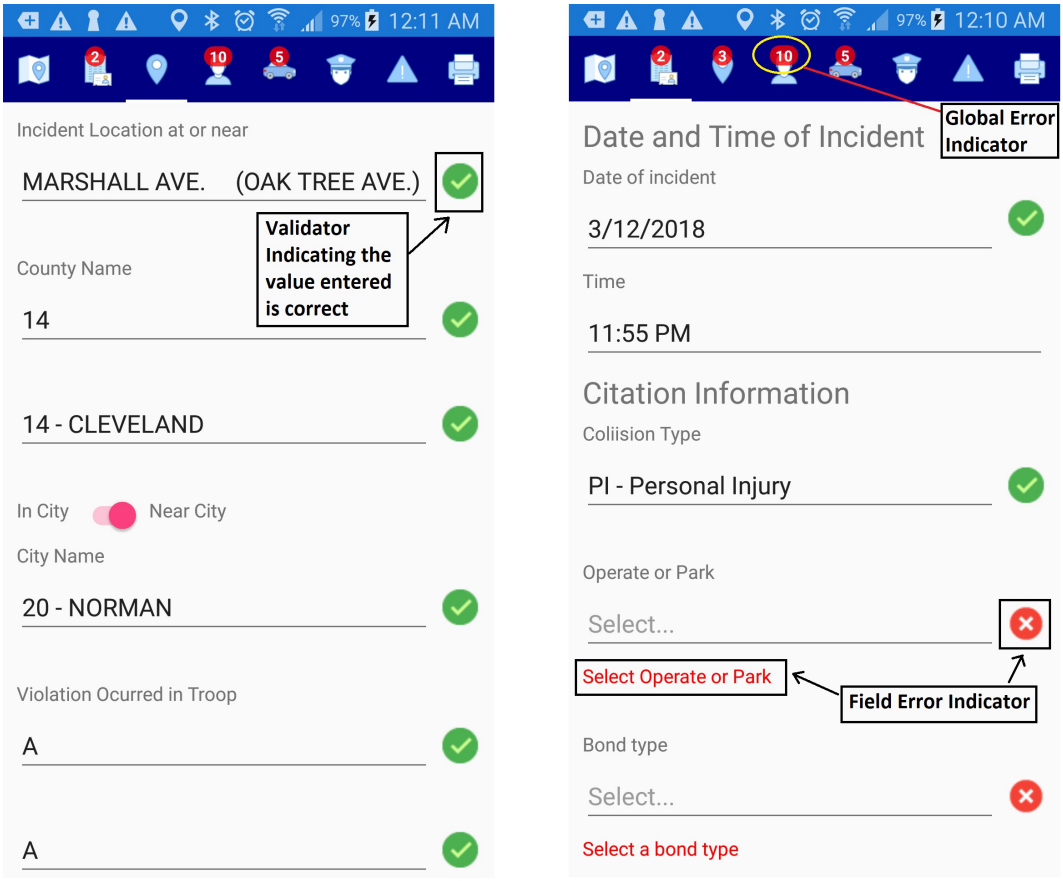


Figure 4.20: Screenshot showing validations on the form.

as we wanted to have a powerful and scalable tool that could be easily reused. The development of custom controls was needed to make the validation error clearly visible because the common controls that come with each operating system do not support validations. This requires the development of custom picker menus and textbox entries. Pickers are pop-up menus that allow the user to pick a value from a list, while textboxes are directly typed input fields. See Figure 4.20. Tabs are used to hold all of the parts of the form. Because of this, it makes sense to place the number of errors in that specific section of the form in the tab headers. See Figure 4.20.

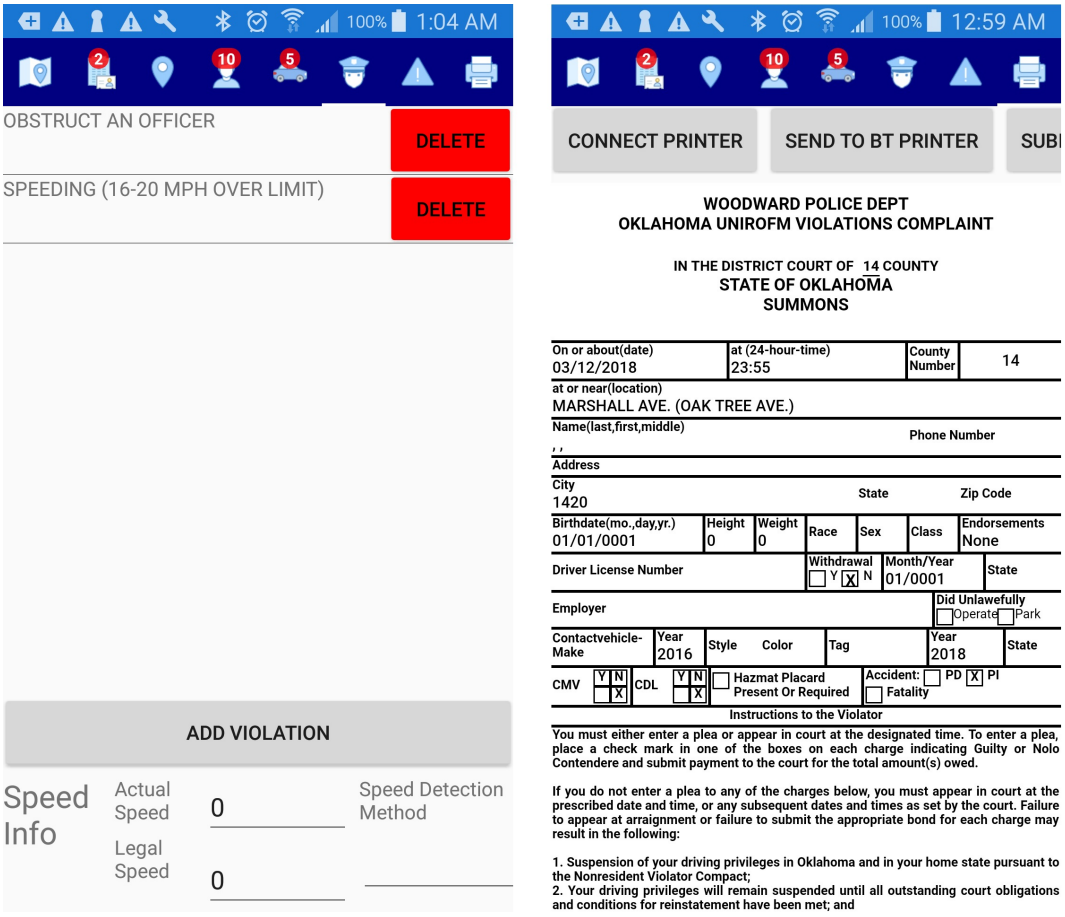


Figure 4.21: Screenshot of a rendered citation form.

4.8 Multiple Violations and Citation Generation

Allowing the officers to add multiple violations and citations is a crucial feature of this project. This way, officers do not have to fill out an entirely new form just to create an additional violation. The interface was developed so that the officer can quickly search within a list of violations and add as many as needed. The first thing the interface displays when the officer attempts to add a violation is a view with all of the possible violations listed and a search bar on the top of the screen. From here, the officer can perform a quick search simply

by entering any word mentioned in the violation and easily select the desired option from the list. As soon as the option is selected, the application brings the user back to the violation view where the officer can delete the violation added or tap to add more detail- such as a remark.

As mentioned in Chapter 3, the citation needs to be generated based on the previous information and violations assigned by the officer. To generate this citation in a way that would duplicate the exact look of the current citation's format, HTML was used and coded from scratch. More specifically, the citation is actually coded using Razor syntax from ASP.NET which is a web framework that uses C# as language. This Razor syntax becomes rendered and is displayed as normal HTML with the actual code resembling HTML with some C# code embedded in it. This allows display values from libraries to be easily assigned and also makes it simple to dynamically render the desired number of citations or violations inserted by the officer. This HTML is later displayed on a web view embedded in the last section of the form. This specific web view control had to be customized and required writing native code in order for it to be possible to print. To do this, the review content is first converted to a JPG image that is later passed to the printer driver which prints the citation. See Figure 4.21.

4.9 Backend

The backend is an Application Interface (API) that is in charge of receiving all of the information from the client application. The API contains all the model definitions for all of the tables. The API has routes that allow HTTP request to insert or read data to each table in the database. The previous

PARIS database was designed based on TraCS and already has all of the fields necessary for the State of Oklahoma. And as it was instructed, PARIS+ follows the same design. The API runs through a Proxy Server provided by the I.T. personnel of ITS. The proxy server allows having Mod Security which is a real-time web application framework that provides an extra layer of security by protecting the server from vulnerability present in web application code, making the API more secure. All the routes of the API, except for the login, are protected. This means that only authenticated request can be processed. The rest of the requests will be replied to with a forbidden error.

4.10 Citation Manager

The citation manager 4.22 is another unique solution for the PARIS+ mobile application that works as a service for the courts in Oklahoma. The system offers the courts access to view the citations generated by the PARIS desktop and PARIS+ mobile application.

4.10.1 Overview

Authentication

Much like PARIS+, the citation manager web application has a token-based authentication with the same approach as the PARIS+ API. The citation manager allows the court administrators to manage their user's accounts where they are able to add or remove users and reset passwords from their administrator menu.



Figure 4.22: Citation Manager sign-in page screenshot.

Search Tool

The citation manager has a sophisticated search tool that allows its user to easily perform a search. It allows the user to set a range of dates for the search using two date pickers at the top of the interface. It also allows users to real-time filter by any field simply by typing in the generic search bar at the top of the screen. Additionally, the search bar supports the use of wildcards that make executing the search far more simple. For example, a user could simply start typing in the beginning of the name of the person he is looking for and append a % at the end, and the search tool will look in all of the fields for any information beginning with that entry. In order to perform a more specific search, the user can append a & at the end of his text and start writing in an additional search parameter such as the city where the incident occurred.

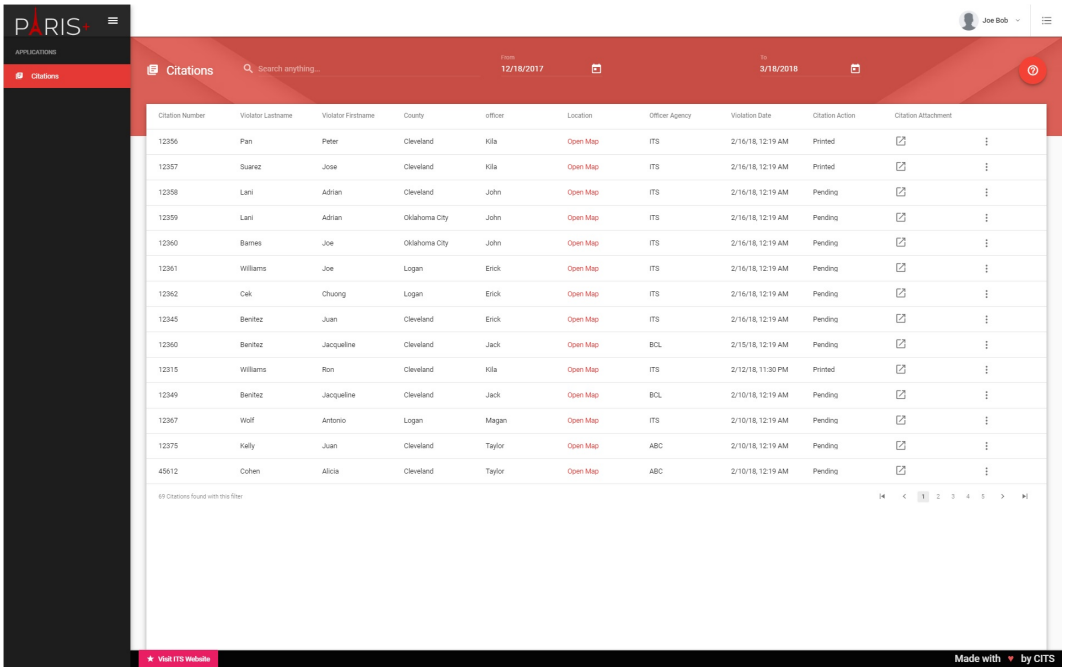


Figure 4.23: Citation Manager dashboard screenshot.

For example, if the user were to type in Rod%&Norman, the search tool would return all of the entries that contain a field that begins with Rod, and another field with the value Norman. The results are displayed in a table format with multiple pages of 14 records each. These records are lazy loaded- meaning that the data is loaded on demand. Therefore, if a search results in many records, the API only returns 14 records per page. This allows the web application to perform well and the API response to remain fast even with a high volume of records.

4.10.2 Data Acquisition

The citation manager receives its data from the PARIS+ API. Once data has been inserted by an officer, the API can then sync that information to the citation manager using its API. This allows other systems, such as PARIS

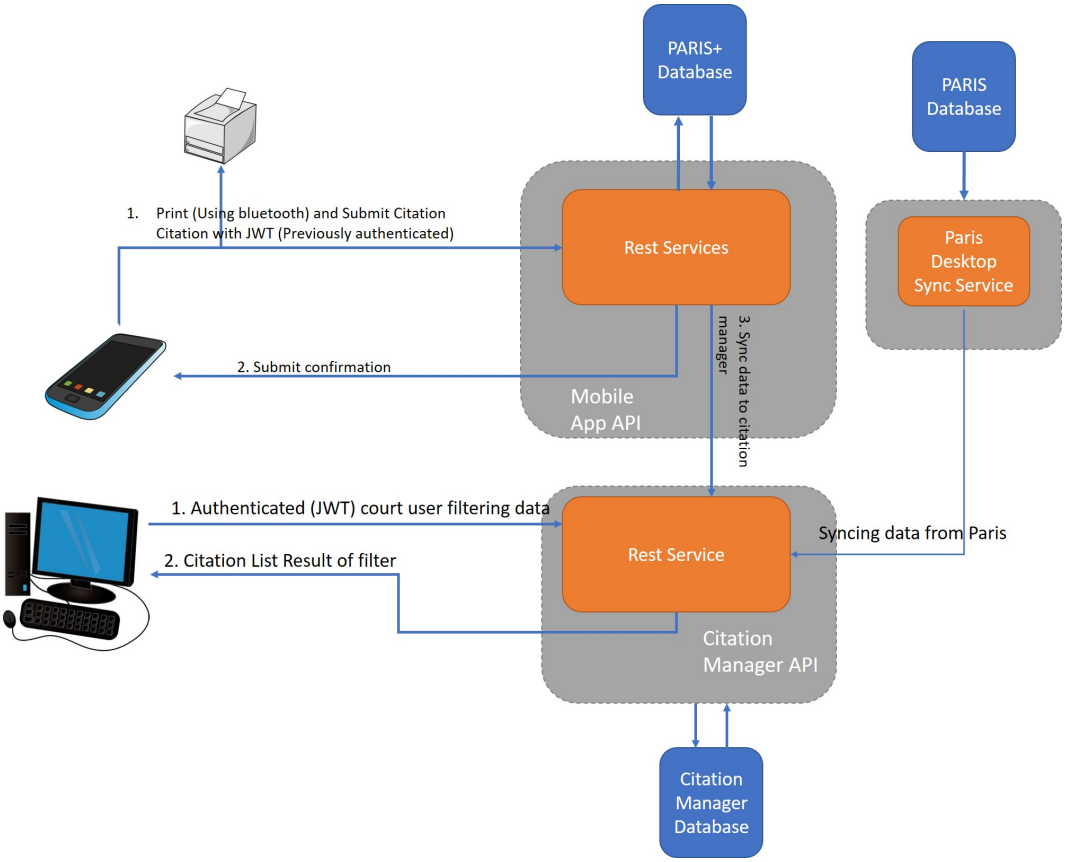


Figure 4.24: Citation Manager block diagram.

Desktop, to sync its data to the citation manager. The system is capable of receiving data from many services. Therefore, in the future, the private API could be opened to other services that would also like to submit data to the Oklahoma court. The block diagram in Figure 4.24 explains how the data is inserted into the citation manager using its API.

4.11 Budget

Throughout the entire process of this project, the fact that this solution needed to be as affordable as possible was always kept in mind. In order to keep

costs within budget, extra effort was needed during development to ensure that the code could function properly when used with expensive hardware such as the previously mentioned thermal printers. To keep costs as low as possible, the final solution was made to be capable of working on the officers' existing smartphones and would only require an external printer which would make handing over a physical copy of the ticket to the violators possible.

Table 4.1 shows the estimated budget for an agency where the officers do not have a smartphone. The one time investment required per user is around $200, which is even less than the typical printer cost (between $300 and $600 depending on the brand, size and portability). This price includes a low range smartphone. The cost to run PARIS per agency is estimated to be around $166 a month or around $2000 a year, regardless of the number of officers.

Table 4.1 shows the estimated budget for an agency where the officers already have smartphones (most cases). The only one-time investment required in this situation is simply the cost of the printer, which is around $71. The total cost to run PARIS per agency is estimated to be around $166 a month or around $2000 a year regardless of the number of officers.

An agency with 10 officers that already possess smartphones could begin using PARIS+ with a total initial investment of less than $2700 in the first year and expect to pay an additional $2000 for each of the following years. Agencies with officers who currently do not possess smartphones could begin with an investment of roughly $4020 in the first year.

This price-point is very promising considering the fact that the current PARIS' required investment for a similar agency for the first year alone is

One Time Expenses Per Officer		
Item	Detail	Price
1	Thermal Printer	$71.30
2	SmartPhone Samsung Galaxy J3*	$129.99
	Total One Time Expenses	$201.29
	** Item 2 is only necessary if officers do not yet have a phone they own*	
	Monthly Cost Per Agency	
Item	**Detail**	**Price**
1	Paris Maintenance	$ 41.66
2	Support Team	$ 41.66
3	Service Fee	$ 83.33
	Total Monthly Cost Per Agency	$ 166.65

Table 4.1: Required budget to use PARIS+ including smartphone.

around $15,000. This cost includes refurbished Toughbook computers, which only account for the hardware requirements needed in order to implement the system in an agency. By removing these requirements, PARIS+ is able to provide a far more affordable solution.

One Time Expenses Per Officer		
Item	Detail	Price
1	Thermal Printer	$ 71.30
	Total One Time Expenses	$ 71.30
Monthly Cost Per Agency		
Item	Detail	Price
1	Paris Maintenance	$ 41.66
2	Support Team	$ 41.66
3	Service Fee	$ 83.33
	Total Monthly Cost Per Agency	$ 166.65

Table 4.2: Required budget to use PARIS+.

Chapter 5

Conclusion and Future Work

In this Thesis, I have shown the details of the design and development of my solution to the lack of electronic submission systems for small police agencies in Oklahoma.

I have explained what electronic citation systems are and why it is important to have accurate data reports from police agencies. Additionally, I reviewed the private and public solutions currently available on the market. Even though they offer solutions to many of the problems that are caused by handwritten citations, they do not solve all of the ones detailed in this Thesis including offering the ability to easily add multiple violations within a citation. I have also presented all of the possible options for developing mobile applications as well as why touchscreen phones have the potential of becoming very useful user input devices for developing this kind of application.

I explained in detail the implementation of the most important parts of the application such as login, barcode scanning, validations, a capability of adding multiple violations, and the citation manager tool for the courts.

The intent of the developed application is to provide a police data collection system that is easy to use, while still gathering high accuracy data and is affordable for most police agencies.

5.1 Contribution

In this thesis, the solution I have presented, as previously explained, is a shared work. In the presented solution, my personal contributions are the following:

- Development of the Application Interface(API) for PARIS+
- Structure and Organization of the Project Libraries
- Login Interface
- Printer Driver
- Citation Page Holder
- Bar Code Scanner Interface
- MVVM Library
- Legacy Dropdown Library
- User Control Library
- PARIS+ Ciration Manager Client and Login
- PARIS+ Citation Manager Search Tool
- Development of the Application Interface(API) for PARIS+ Citation Manager Tool

5.2 Future Work

While the application development turned out to be a success, as all of the set goals where reached, improvements can still be made. One important task that is still left to be done is to migrate the current libraries to the newest libraries on Xamarin Forms. The project is currently using Portable Class Libraries (PCL) which have already been surpassed by the new .Net Standard Libraries that have been released. However, this transition is expected to be relatively straight-forward and could be completed in the near future.

Another feature that still needs to be completed is the offline login and the data synchronization. Although this feature is currently partially working, it needs to be improved and tested to ensure that it is working correctly .

Additionally, some dependency services for iOS need to be polished and finished. These include the Location Lookup and the Printing Service. While these services have been tested and are working perfectly on Android, as this was the priority, the iOS implementation still needs to undergo some improvements.

The application does still need to be tested with a beta group to ensure that it works perfectly for the officers. To do this, the beta group will be testing the printers to measure how long they can survive in rough environments and determining whether the overall application fulfills the officers' needs and expectations.

After a beta program has been run, the application will have to be deployed to many agencies using the Business to Business (B2B) services provided by both Apple and Android. This will make the update process, maintenance,

and adding of agencies both simple and affordable. This application was developed with the ability to scale. Because of this, Crash, Stored Vehicles, and DUI Forms could easily be implemented and added to the application to provide even more services to small agencies in the future.

Bibliography

[1] "Zebra Printers | Desktop, Mobile, Industrial and more." [Online]. Available: http://www.zebra.com/us/en/products/printers.html

[2] AAMVA, "DL ID Card Design Standard," 2016. [Online]. Available: https://www.aamva.org/dl-id-card-design-standard/

[3] M. A. Abdel-Aty and A. E. Radwan, "Modeling traffic accident occurrence and involvement," *Accident Analysis & Prevention*, vol. 32, no. 5, pp. 633–642, Sept. 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0001457599000949

[4] Adam Honeyman, "Meeting about The usage of electronic citation on Woods County Sheriff Department," Mar. 2018, private Communication.

[5] Adrian Campbell, "History of ITS and electronic citation software," Mar. 2018, private Communication.

[6] P.-A. Albinsson and S. Zhai, "High Precision Touch Screen Interaction," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '03. New York, NY, USA: ACM, 2003, pp. 105–112. [Online]. Available: http://doi.acm.org/10.1145/642611.642631

[7] Alex Schoof, "What is an SDK? (C++) - Stack Overflow." [Online]. Available: https://stackoverflow.com/questions/1291226/what-is-an-sdk-c/1291304

[8] Alicia Keys, "Collision Report from SAFE-T," 2018, personal Comunication.

[9] N. Aljeri and A. Boukerche, "A predictive collision detection protocol using vehicular networks," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct. 2017, pp. 1–5.

[10] Altex Soft Inc, "Pros and Cons of Xamarin vs Native Mobile Development," Nov. 2017. [Online]. Available: https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/

[11] Apple, "Apple at Work - Technology designed for all the ways your employees want to work." [Online]. Available: https://www.apple.com/business/products-platform/

[12] ——, "Education - Ignite the creativity in every student." [Online]. Available: https://www.apple.com/education/

[13] Apple Inc, "iOS 11.1 SDK Release Notes," Oct. 2017. [Online]. Available: https://developer.apple.com/library/content/releasenotes/General/RN-iOS-11.1/index.html

[14] Artyom Dogtiev, "App Download and Usage Statistics 2017," Jan. 2018. [Online]. Available: http://www.businessofapps.com/data/app-statistics/

[15] H. Avsar, J. E. Fischer, and T. Rodden, "Designing touch screen user interfaces for future flight deck operations," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, Sept. 2016, pp. 1–9.

[16] Bloomberg, "Alibaba.com Ltd: Company Profile." [Online]. Available: http://www.bloomberg.com/profiles/companies/1688:HK-alibaba-com-ltd

[17] D. Britch, U. Brad, C. Petzold, and C. Dunn, "Introduction to DependencyService - Xamarin," June 2017. [Online]. Available: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/dependency-service/introduction

[18] Brother Printers, "Brother Printers Laser, Inkjet & All-in-Ones." [Online]. Available: http://www.brother-usa.com/printer/

[19] N. Brown, "Why the industrial enterprise is moving to iOS," Mar. 2018. [Online]. Available: https://www.cio.com/article/3263400/operating-systems/why-the-industrial-enterprise-is-moving-to-ios.html

[20] A. Burns, B. Umbaugh, and C. Dunn, "Introduction to Portable Class Libraries - Xamarin," Mar. 2017. [Online]. Available: https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/pcl

[21] Canonical Ltd., "SDK - the Ubuntu SDK | Ubuntu Phone documentation." [Online]. Available: https://docs.ubuntu.com/phone/en/platform/sdk/

[22] CB Insights Research, "Apple Is Going After The Health Care Industry, Starting With Personal Health Data," Sept. 2017. [Online]. Available: /research/apple-health-care-strategy-apps-expert-research/

[23] Center for Intelligent Transportation Systems, "Oklahoma TraCS Pilot Project," Oklahoma, Tech. Rep., Feb. 2008.

[24] Chuong Nguyen, "History of Electronic Citations in Oklaoma," Mar. 2018.

[25] A. Cockburn, D. Ahlstrm, and C. Gutwin, "Understanding performance in touch selections: Tap, drag and radial pointing drag with finger, stylus and mouse," *International Journal of Human-Computer Studies*, vol. 70, no. 3, pp. 218–233, Mar. 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1071581911001546

[26] Committee for the Study of Strategic Transportation Data Needs, Transportation Research Board, and National Academies of Sciences, Engineering, and Medicine, *Data for Decisions: Requirements for National Transportation Policy Making: Requirements for National Transportation Policy Making – Special Report 234*. Washington, D.C.: Transportation Research Board, Jan. 1992. [Online]. Available: https://www.nap.edu/catalog/11407

[27] Daniel Witte and Philipp von Weitershausen, "React Native for Android: How we built the first cross-platform React Native app." [Online]. Available: https://code.facebook.com/posts/1189117404435352/react-native-for-android-how-we-built-the-first-cross-platform-react-native-app/

[28] Danny Ryback, "R.I.P. the Native Mobile App, 2008-2017 Leo Burnett." [Online]. Available: https://apple.news/AeRF-cujfQrayjVMeMCoUwA

[29] Drifty, "Welcome to Ionic - Ionic Framework." [Online]. Available: https://ionicframework.com/docs/v1/guide/preface.html

[30] eCitationCoalition, "Coalition Members." [Online]. Available: http://www.ecitationcoalition.com/coalitionmembers/

[31] ——, "Guidelines for Evaluating and Implementing eCitation Systems," Dec. 2016. [Online]. Available: http://www.ecitationcoalition.com/wp-content/uploads/2016/12/eCitation-Coalition-White-Paper-Guidelines-for-Evaluating-and-Implementing-eCitation-Systems.pdf

[32] Eric Cannaday, "History of Electronic Citations in Oklahoma," Apr. 2018, private communication.

[33] Eric Fultz and David Crist, "Still writing citations by hand?" Mar. 2016. [Online]. Available: https://www.officer.com/command-hq/technology/traffic/article/12165552/still-writing-citations-by-hand

[34] O. Ethelbert, F. F. Moghaddam, P. Wieder, and R. Yahyapour, "A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications," in *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug. 2017, pp. 47–53.

[35] Facebook Ink., "React Native A framework for building native apps using React." [Online]. Available: https://facebook.github.io/react-native/index.html

[36] ——, "React Native A framework for building native apps using React." [Online]. Available: https://facebook.github.io/react-native/

[37] Fixed Technology LLC, "Traffic Ticket Lawyer." [Online]. Available: https://www.fixedlaw.com

[38] B. L. Foster, "Married to Their Smartphones (Oh, and to Each Other, Too)," *The New York Times*, Oct. 2016. [Online]. Available: https://www.nytimes.com/2016/10/30/style/smartphones-iphone-marriage-husbands-wives-technology.html

[39] Google, "How to download Android Studio and SDK Tools." [Online]. Available: https://developer.android.com/studio/index.html

[40] IDC, "IDC: Smartphone Vendor Market Share." [Online]. Available: https://www.idc.com/promo/smartphone-market-share/vendor

[41] Jason Culliton, "Thermal Barcode Printing," Oct. 2011. [Online]. Available: https://www.l-tron.com/thermal-barcode-printing/

[42] Jim Nielsen, "Discretion - The Art of Law Enforcement," June 2011. [Online]. Available: http://www.rasmussen.edu/degrees/justice-studies/blog/discretion-the-art-of-law-enforcement/

[43] Z. X. Jin, T. Plocher, and L. Kiff, "Touch Screen User Interfaces for Older Adults: Button Size and Spacing," in *Universal Acess in Human Computer Interaction. Coping with Diversity*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, July 2007, pp. 933–941. [Online]. Available: https://link-springer-com.ezproxy.lib.ou.edu/chapter/10.1007/978-3-540-73279-2_104

[44] Jon Fingas, "Microsoft officially ends support for Windows Phone," July 2017. [Online]. Available: https://www.engadget.com/2017/07/11/windows-phone-support-ends/

[45] Jonathan Creamer, "Why would you NOT use TypeScript?" Feb. 2018. [Online]. Available: http://jonathancreamer.com/why-would-you-not-use-typescript/

[46] Joseph P. Havlicek, "Meeting about History of Electronic Citations in Oklahoma," Mar. 2018, private Communication.

[47] Joseph P. Havlicek and Oklahoma Highway Safety Office Contract, "OKLAHOMA HIGHWAY SAFETY OFFICE CONTRACT," Mar. 2010.

[48] Kristopher Sandoval, "What is the Difference Between an API and an SDK? | Nordic APIs |," June 2016. [Online]. Available: https://nordicapis.com/what-is-the-difference-between-an-api-and-an-sdk/

[49] O. Le Goaer and S. Waltham, "Yet Another DSL for Cross-platforms Mobile Development," in *Proceedings of the First Workshop on the Globalization of Domain Specific Languages*, ser. GlobalDSL '13. New York, NY, USA: ACM, 2013, pp. 28–33. [Online]. Available: http://doi.acm.org/10.1145/2489812.2489819

[50] S. Lee and S. Zhai, "The Performance of Touch Screen Soft Buttons," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 309–318. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518750

[51] Leonid Bershidsky, "Smartphone Addiction Is a Problem Apple Won't Solve," *Bloomberg.com*, Jan. 2018. [Online]. Available: https://www.bloomberg.com/view/articles/2018-01-09/smartphone-addiction-is-a-problem-apple-won-t-solve

[52] X. Li, D. Chang, H. Pen, X. Zhang, Y. Liu, and Y. Yao, "Application of MVVM design pattern in MES," in *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, June 2015, pp. 1374–1378.

[53] Mark Gurman, "Apple Launches USD299 iPad for Education," *Bloomberg.com*, Mar. 2018. [Online]. Available: https://www.bloomberg.com/news/articles/2018-03-27/apple-launches-low-cost-ipad-for-education-targeting-google

[54] D. Meyers, "TraCS National Model Overview," June 2014. [Online]. Available: https://www.pcb.its.dot.gov/t3/s140612/s140612_TIM_perf_measures_presentation_meyers.pdf

[55] Microsoft, "Build a Native Android UI & iOS UI with Xamarin.Forms - Xamarin." [Online]. Available: https://www.xamarin.com/forms

[56] ——, "Windows SDK archive - Windows app development." [Online]. Available: https://developer.microsoft.com/en-us/windows/downloads/sdk-archive

[57] M. Mller, P. Nadarajan, M. Botsch, W. Utschick, D. Bhmlnder, and S. Katzenbogen, "A statistical learning approach for estimating the reliability of crash severity predictions," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2016, pp. 2199–2206.

[58] National Model, "National Model - Home." [Online]. Available: http://www.teginc.com/nationalmodel/

[59] NativeScript, "NativeScript." [Online]. Available: https://www.nativescript.org/

[60] A. K. Orphanides and C. S. Nam, "Touchscreen interfaces in context: A systematic review of research into touchscreens across settings, populations, and implementations," *Applied Ergonomics*, vol. 61, pp. 116–143, May 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003687017300212

[61] Paul Dixon, "What is TypeScript and why would I use it in place of JavaScript? - Stack Overflow," Oct. 2012. [Online]. Available: https://stackoverflow.com/questions/12694530/what-is-typescript-and-why-would-i-use-it-in-place-of-javascript/12694578

[62] A. Popescu, "Keep Your Head Up: How Smartphone Addiction Kills Manners and Moods," *The New York Times*, Jan. 2018. [Online]. Available: https://www.nytimes.com/2018/01/25/smarter-living/bad-text-posture-neckpain-mood.html

[63] Quicket Solutions, "Quicket Home Page." [Online]. Available: https://www.quicketsolutions.com/qs/

[64] C. P. R. Raj and S. B. Tolety, "A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach," in *2012 Annual IEEE India Conference (INDICON)*, Dec. 2012, pp. 625–629.

[65] P. Reichelt, "Why I left Xamarin behind in favor of React Native," Nov. 2016. [Online]. Available: https://medium.com/@reicheltp/dev-diary-4-why-i-left-xamarin-behind-35817ccd07b2

[66] D. Robinson, "Exploring the State of Mobile Development with Stack Overflow Trends," May 2017. [Online]. Available: https://stackoverflow.blog/2017/05/16/exploring-state-mobile-development-stack-overflow-trends/

[67] C. Rong, L. Zhen-ya, J. Yan-hu, Z. Yi, and T. Li-yu, "Coding principle and implementation of two-dimensional PDF417 bar code," in *2011 6th IEEE Conference on Industrial Electronics and Applications*, June 2011, pp. 466–468.

[68] Saltus Technologies, "digiTicket Key Features." [Online]. Available: http://www.saltustechnologies.com/key-features

[69] SceneDoc, "SceneDoc 'Mobile-First' Platform for Public Safety." [Online]. Available: https://scenedoc.com/

[70] A. Sears and B. Shneiderman, "High precision touchscreens: design strategies and comparisons with a mouse," *International Journal of Man-Machine Studies*, vol. 34, no. 4, pp. 593–613, Apr. 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0020737391900378

[71] Sonya Wolff, "History of Electronic Citations in Oklahoma," Mar. 2018, private Communication.

[72] Star, "Star Micronics." [Online]. Available: http://www.starmicronics.com/

[73] Statista, "Global smartphone market share 2017." [Online]. Available: https://www.statista.com/statistics/271496/global-market-share-held-by-smartphone-vendors-since-4th-quarter-2009/

[74] TBL Systems, Inc., "iCitation/eCitation." [Online]. Available: https://www.thinbluereports.com/tbl-products-law-enforcement/software-app/icitationecitation/

[75] Technical Activities Division, Transportation Research Board, and National Academies of Sciences, Engineering, and Medicine, *Integrating Roadway, Traffic, and Crash Data: A Peer Exchange*. Washington, D.C.: Transportation Research Board, Jan. 2006. [Online]. Available: https://www.nap.edu/catalog/23214

[76] The Office of Justice Programs, Bureau of Justice Assistance, and The Department of Transportation, "The Use of Electronic Citations: A Nationwide Assessment," The Office of Justice Programs, Tech. Rep., June 2003. [Online]. Available: https://it.ojp.gov/documents/20030619_BJA_Study_of_Electronic_Citations.pdf

[77] Tyler Technologies, "E-Citation Software." [Online]. Available: https://www.tylertech.com/solutions-products/brazos-product-suite

[78] W3C, "Web Authentication: An API for accessing Public Key Credentials - Level 1." [Online]. Available: https://w3c.github.io/webauthn/

[79] T. Warren, "Windows Phone dies today," July 2017. [Online]. Available: https://www.theverge.com/2017/7/11/15952654/microsoft-windows-phone-end-of-support

[80] H. Woo, Y. Ji, H. Kono, Y. Tamura, Y. Kuroda, T. Sugano, Y. Yamamoto, A. Yamashita, and H. Asama, "Lane-Change Detection Based on Vehicle-Trajectory Prediction," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1109–1116, Apr. 2017.

[81] Zebra Technologies, "Direct Thermal and Thermal Transfer Printing." [Online]. Available: http://www.zebra.com/us/en/resource-library/getting-started/direct-thermal-thermal-transfer.html

[82] K. Zheng and W. Jiang, "A token authentication solution for hadoop based on kerberos pre-authentication," in *2014 International Conference on Data Science and Advanced Analytics (DSAA)*, Oct. 2014, pp. 354–360.