TagIT: The Checked Item Solution

by

James Craig Slusher, Aaron Westermeyer, & George Santen

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

_____ James Craig Slusher _____	_____04/14/2019_____
James Craig Slusher	Date
_____Aaron Westermeyer _____	_____04/14/2019_____
Aaron Westermeyer	Date
_____George Santen_____	_____04/14/2019_____
George Santen	Date
_____Abdou Fall_____	_____04/14/2019_____
Abdou Fall, Faculty Advisor	Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2019

TABLE OF CONTENTS

Section **Page**

List of Illustrations

TABLES

FIGURES

ACRONYMS AND ABBREVIATIONS

API	Application Programming Interface.
ASP.NET	An open source Web framework for building modern Web apps and services with .NET.
AWS	Amazon Web Services
QR Code	Quick Response Code. A type of matrix barcode first designed in 1994 for the Japanese automotive industry.
SQL	Structured Query Language. A standardized query language for requesting information from a database.

ABSTRACT

Current claim-check technology relies on a numbered, two-part paper ticket. But, according to the New York Times, anywhere from 5 to 20 percent of patrons who check items, lose their tickets (Bruni, 2018). TagIT will replace these fragile paper tickets and streamline the item check-in process using common handheld devices and wireless access to cloud computing. With TagIT, an attendant will create a unique record for each patron, adding images of the checked items using a smartphone or tablet's on-board camera. The attendant will select the item image and scan the storage bin Quick Response Code, or QR code, in which the item is stored. When all items have been imaged and added to the record, the patron will receive an E-mail with a Web link to the record. The images, QR codes, and bin identification are now part of the patron's record and can be viewed and verified using the link. The link will also provide a means to request the return of the checked items. And if a patron forgets to retrieve an item, TagIT will provide the means to contact the patron with retrieval instructions.

1. PROBLEM STATEMENT

1.1 Introduction

People regularly attend events and need a dependable system to ‘check’ or store items. The applications found in the App Store are targeting individual users wishing to manage their checked items, not industry or business users. A dependable electronic checked item tracking system has not been available for most organizations.

1.2 Project Description

The purpose of this project was to replace the traditional paper check-ticket to an electronic system. TagIT will concentrate on providing the business operator with a simple and affordable, yet powerful checked-item tracking system. TagIT will supply event organizers and businesses the tools to provide patrons with secure checked item storage, utilizing standard electronic handheld devices such as smart phones and tablets, or personal computers. The item might be musical instruments for a band between sets at a festival or any item that needs to be stored securely. Security requirements vary in each case, but each unique item must be tracked correctly. In cases requiring the highest security, the need to track each item further requires a demonstrable ‘chain of possession, ensuring proper handling together with access logging. TagIT will scale from simple needs to more complex requirements based on organizational needs. TagIT will provide the tools to accomplish these tasks using mobile devices such as tablets or smart phones, as well as laptop or desktop PC’s outfitted with the necessary ‘off-the-shelf’ hardware and cross-platform Web browsers.

1.3 Problem

Current operations typically rely on paper claim checks or tickets. According to the New York Times, from 5 to 20 percent of patrons who check items, lose their tickets.

CheckMyCoat, a competing, user-oriented application, depends on businesses 'partnering' with the application provider to enable the coat-check tracking.

Another competing product, CoatChex, is a business-oriented system. And while the CoatChex application has good business class options, it requires expensive, proprietary hardware to manage the checked items.

1.4 Solution

TagIT will be able to identify, track, store, and return property or items of interest. An event organizer can image checked items, collect consumer contact information, provide the consumer with storage details, verify item security, deliver the correct checked items, and optionally store or clear the consumer information.

The product has been developed for mobile platforms such as smart phones and tablets, using cross-platform Web browsers. Data storage options will include local storage for simple installations or cloud hosting utilizing AWS for more robust installations.

TagIT will image each item using a mobile device on-board camera or a camera attached to a PC, assign a unique ID to each item, and provide user access to the data.

Optionally, a repeat customer can opt for a unique personal QR code to facilitate quick checking and recovery of items. The business might use this to market to their 'best' customers, with discounts, offers, or coupons.

1.5 User Profile

The user profile description will be used through the development of the product. It will provide the team with details of the target audience or market for whom the product is being created. Throughout development, the team will revisit the user profile to ensure the real reason for this project, the users, remains the focus.

1.5.1 Project Title

TagIT: The Checked Item Solution

1.5.2 Potential Users

Admin User - for system configuration

Attendant - individuals checking items for event organizers

1.5.3 Software, Interface, and Related Experience

Attendant - This project will be primarily targeted towards technologically competent individuals that will most likely have experience with the type of hand-

held devices used with the system. Operation activities will include use of browsers, integrated cameras, and imaging devices.

Users not from the technology sector or that lack a technical background will still be able to use the product with minimal instruction.

Regardless of the user's background, this project will ensure a seamless transition into the creation of user records, imaging checked items, and return of the correct item to the user.

1.5.4 Experience with Similar Applications

Administrator - The initial system configuration will require a level of technical competence of the individuals setting up the system. This may be through typical use of browser-based applications and forms, such as on-line retailing applications. A setup 'Wizard' will be available to assist in the initial configuration tasks.

Attendant - It is unlikely most users will have had experience operating a system like TagIT. However, design of the product is based upon a user's ability to operate it with minimum training requirements. Operation will be like other browser-based applications.

1.5.5 Task Experience

Event organizers may have differing requirements according to the needs of individual events. The system allows for customization of storage details, including storage types (bins, rooms, hangars, etc.), notification methods, and identification details.

1.5.6 Frequency of Use

Administrator - Configuration of this product will typically occur once.

Attendant - For demonstration and actual use, the product will be used continuously throughout the day. At demonstration, there will be several scenarios present to create records using sample devices.

1.5.7 Key Project Design Requirements:

- Browser-based system
- Visually fluid and easy to navigate UI
- Ability to customize configuration of storage details
- Automated customer notifications and information
- Tips to get started
- Quick start/setup

1.6 Use Case Diagram

Figure 1; Use Case Diagram presents the planned use case diagram for this project. This represents the base level of requirements we have for our product. After these requirements have been met, more features may be added to improve the customer experience.

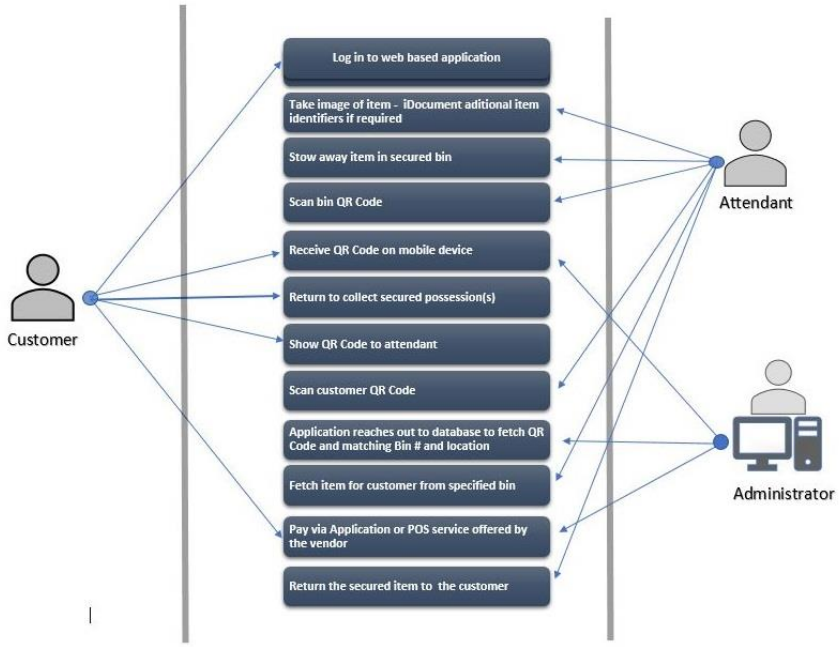


Figure 1; Use Case Diagram

2.1.1 Budget (projected)

As shown in Table 1; Project Budget - Projected, estimates are for time (labor) and materials (hardware and software). It was anticipated that using the Sandbox virtual systems provided by CECH would reduce material costs to $0. However, we encountered problems implementing TagIT on Sandbox during the Spring semester. This forced us to investigate alternative platforms, such as local operation and AWS in the cloud.

The development tools, Visual Studio and MS SQL Server were also provided by CECH and were no cost. Labor and hardware costs were estimated based on typical industry rates in effect at the time of the project. However, the actual costs for the project were $0 as the project is part of the degree program requirements for a Bachelor of Science, Information Technology at the University of Cincinnati.

CATEGORY	ITEM	DESCRIPTION	EXPECTED COST	ACTUAL COST
Materials	Hardware	Computers, servers and devices used to create, test, run, and deploy the application.	$4500	<$100
	Software	Frameworks and development tools used to develop the application. (MS Visual Studio 2017)	$1600	$0
Labor	Conferences and Meetings	Funding for employee collaboration, learning events, and meetings.	$450	$25
	Simulated Wage Costs	The predicted wages that would be distributed if the project were done in the current market. (We assumed 3 employees @ $37 USD / hour).	$15000	$0
TOTALS			$21,550	$125

Table 1; Project Budget - Projected

2.1.2 Budget (final labor)

Table 2; Project Budget – Final Labor, shown below, captures the actual labor expenses this project would have accrued in a real business.

4	Westermeyer	Login Page	3		
5		Create Record	6		
6		Place Item	6		
7		Manage Item	1		
8		Email	3		
9		Retrieve Item	5		
10		Testing	9		
11		Launch Application	10		
12		Research	30		
13				101	
14	Santen	Use Case Diagram	3		
15		Network Diagram	2		
16		TagIT Project Poster	10		
17		Cloud Infrastructure Researc	14		
18		UC CECH Sandbox Trial	6		
19		System Testing Template	3		
20		Amazon Web Services	12		
21		GroupMe Remote Meetings	28		
22		Powerpoint Presentation Re	4		
23		Elevator Pitch	2		
24		Speech Prep	4		
25		Final Presentation	6		
26		2019 IT Expo Planning and Ex	12		
27				98	
28	Slusher	Team Contract	3		
29		Requirements	15		
30		Project Plan	5		
31		Project Abstract	7		
32		User Profile	3		
33		Use Case	4		
34		Elevator Speech	3		
35		Poster	2		
36		Presentation	7		
37		Final Report	45		
38		Development	12		
39				106	
40			Avg Hourly Rate		$ 37.00
41			Total	305	$11,285.00

Table 2; Project Budget – Final Labor

2.2 Objectives/Deliverables

The project deliverables were developed from the requirements research conducted at the start of the project. These evolved during the project into the list shown in Table 2; Project Deliverables.

User 'Front-End"
• User information and ID • Item information and ID • Item image and ID
Back-End 'Configuration'
• Configure storage • Add/manage users • Add/manage customers • Add/manage business rules • Payments and Tips • E-mail / Text / Notifications
Database Schema (SQL)
• Develop Structure • Add indexes and keys • Test functionality

Table 3; Project Deliverables

2.3 Milestones

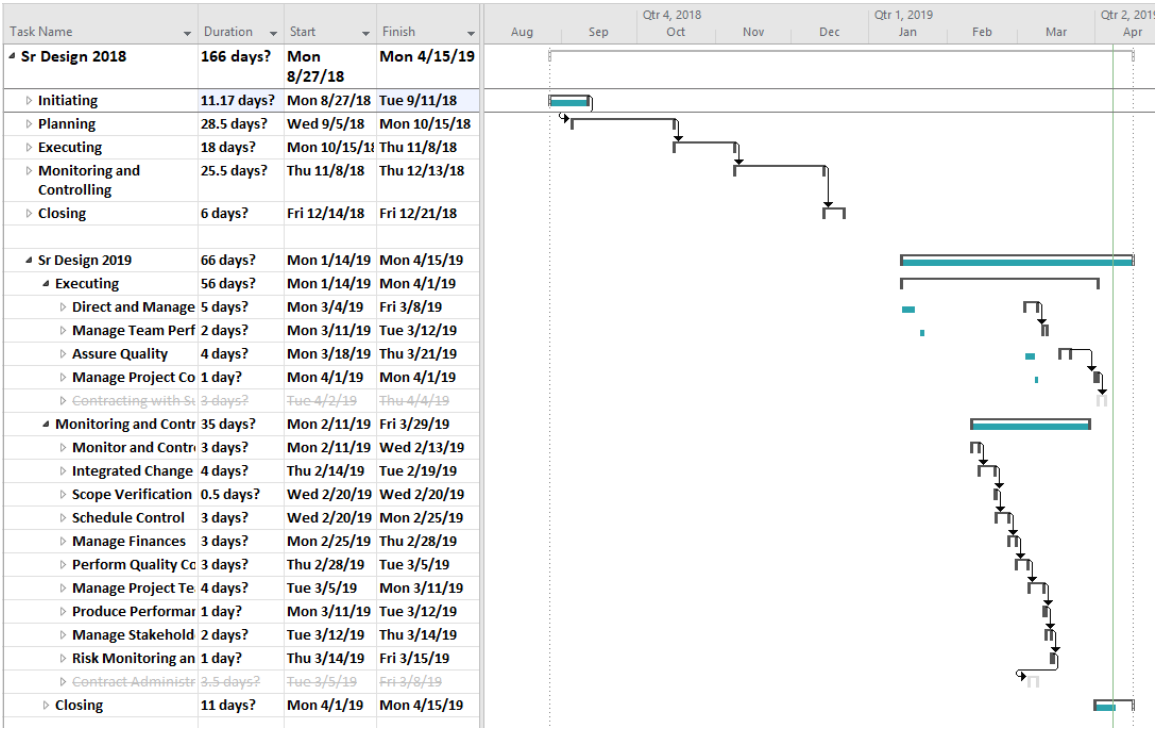
Table 3; Project Milestones, below, shows the anticipated milestones as well as their completion dates.

MAJOR PROJECT MILESTONES (DELIVERABLES)			
FALL OF 2018 MILESTONES			
Home Page Milestone	**10/30/2018**	**Login Page Milestone**	**10/30/2018**
SQL Schema Milestone	**11/16/2018**	**Test PC Setup**	**11/16/2018**
SPRING OF 2019 MILESTONES			
Record Entry Milestone	**2/5/19**	**IIS Deploy Milestone**	**3/15/19**
System Testing Milestone	**3/5/19**	**UI Testing Milestone**	**3/27/19**

Table 4; Project Milestones

2.4 Project Schedule

Figure 2; 2018-2019 Project Schedule, shown below, was the full project schedule for 2018-2019.



Task Name	Duration	Start	Finish
Sr Design 2018	166 days?	Mon 8/27/18	Mon 4/15/19
Initiating	11.17 days?	Mon 8/27/18	Tue 9/11/18
Planning	28.5 days?	Wed 9/5/18	Mon 10/15/18
Executing	18 days?	Mon 10/15/18	Thu 11/8/18
Monitoring and Controlling	25.5 days?	Thu 11/8/18	Thu 12/13/18
Closing	6 days?	Fri 12/14/18	Fri 12/21/18
Sr Design 2019	66 days?	Mon 1/14/19	Mon 4/15/19
Executing	56 days?	Mon 1/14/19	Mon 4/1/19
Direct and Manage	5 days?	Mon 3/4/19	Fri 3/8/19
Manage Team Perf	2 days?	Mon 3/11/19	Tue 3/12/19
Assure Quality	4 days?	Mon 3/18/19	Thu 3/21/19
Manage Project Co	1 day?	Mon 4/1/19	Mon 4/1/19
~~Contracting with St~~	~~3 days?~~	~~Tue 4/2/19~~	~~Thu 4/4/19~~
Monitoring and Contr	35 days?	Mon 2/11/19	Fri 3/29/19
Monitor and Contr	3 days?	Mon 2/11/19	Wed 2/13/19
Integrated Change	4 days?	Thu 2/14/19	Tue 2/19/19
Scope Verification	0.5 days?	Wed 2/20/19	Wed 2/20/19
Schedule Control	3 days?	Wed 2/20/19	Mon 2/25/19
Manage Finances	3 days?	Mon 2/25/19	Thu 2/28/19
Perform Quality Co	3 days?	Thu 2/28/19	Tue 3/5/19
Manage Project Te	4 days?	Tue 3/5/19	Mon 3/11/19
Produce Performar	1 day?	Mon 3/11/19	Tue 3/12/19
Manage Stakehold	2 days?	Tue 3/12/19	Thu 3/14/19
Risk Monitoring an	1 day?	Thu 3/14/19	Fri 3/15/19
~~Contract Administr~~	~~3.5 days?~~	~~Tue 3/5/19~~	~~Fri 3/8/19~~
Closing	11 days?	Mon 4/1/19	Mon 4/15/19

Figure 2; 2018-2019 Project Schedule

3. Technical Elements

3.1 Infrastructure

The back-end infrastructure designed for the TagIT application was modeled after a conventional Database-Web Application. The SQL Server and IIS machines were given static IP addresses within the same range and resided on the same network. Afterwards the team configured the network based on fundamental, and simple server configurations. While there are a variety of platforms to establish a Database-Web Server network, most of them follow the same general layout. The Web Server was configured with Active Directory, IIS, and RSAT using the server manager. For the database server, Microsoft SQL Server 2016 was acquired through the UC student software center and installed on the machine by mounting the provided .iso file. Both of the servers were then set up within the same Active Directory Domain/ Domain controller. Successfully communication between the two servers was established and verified. From that point forward, several workstations and users were created within the Active Directory Domain. This allowed that the rest of the team to access the network using their unique credentials. The server environment provisioning was monitored and adjusted based on the changing needs of the development team.

3.2 Database

The data created by the application was stored in a SQL Server database. SQL was chosen since it is widely supported and familiar to a wide range of companies. This allows TagIt to be implemented without the need to purchase additional software. The curriculum of the software development track meant the team was also very comfortable

with SQL Server, making it the ideal solution for TagIT. The development team also appreciated the fact that SQL Server was constantly being patched and updated, guaranteeing customers would have a quality, well supported database available.

3.3 Application

The application was developed using ASP.NET C# and leveraging Bootstraps front-end strengths. ASP.NET was chosen due to its ease of use and compatibility with SQL Server. Bootstrap was chosen for its position as a mobile-first front-end web development package, making it ideal for designing a mobile friendly application, such as TagIT. User data records will be stored in a SQL Server database. Both ASP.NET and Bootstrap have a plethora of documentation as well, allowing for ease of development and rapid troubleshooting.

3.4 Security

Members of the team have continuously discussed and researched potential risks with the projected and potential applications for TagIT. Due to the lack of a ‘public-facing’ application interface, security risks are greatly reduced. Authorized users will authenticate both to the device (smartphone, tablet, or PC) and within the TagIT system. Active Directory users and groups have been individually assigned their own level of permissions on the network.

4. Visual Elements

4.1 Screen Captures

4.1.1 Login View

Figure 3; Desktop Login, shown below, is representative of the TagIT login screen from a desktop.

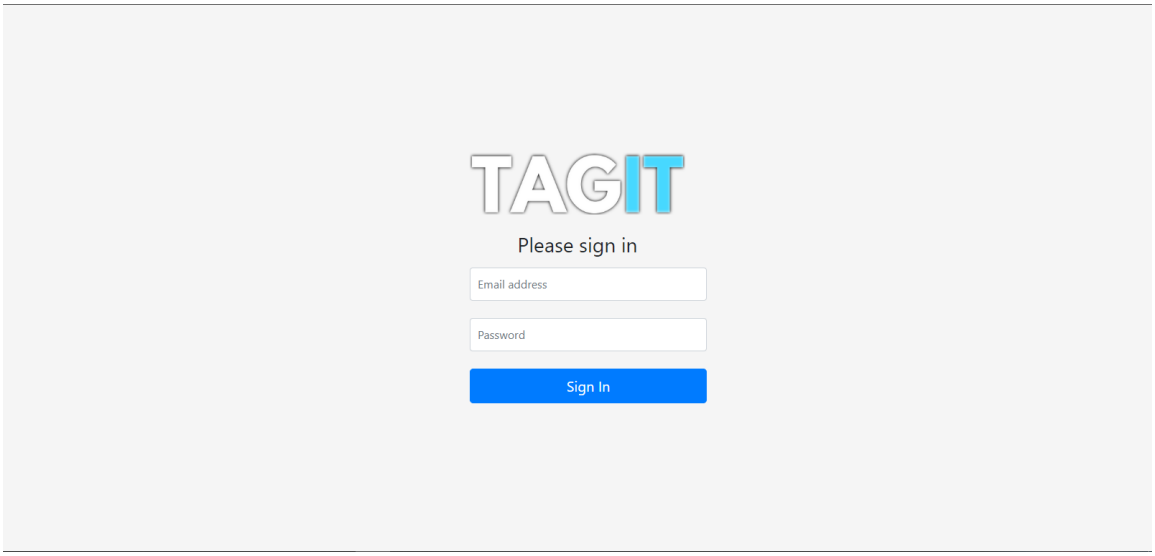


Figure 3; Desktop Login

Figure 4; Mobile Login, shown below, is representative of the TagIT login screen from a desktop.

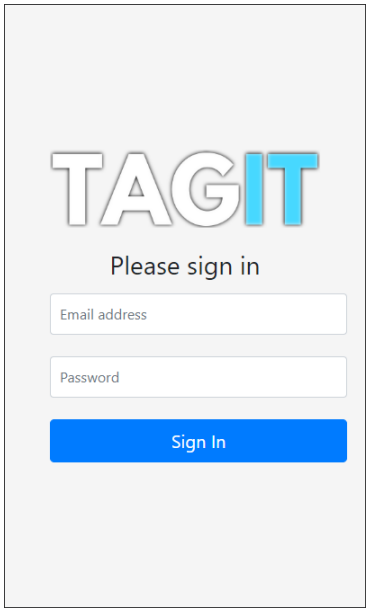


Figure 4; Mobile Login

The login screen requires a username and password for attendants and administrators to operate or modify the system.

4.1.2 Create Record

Figure 5; Create Record Desktop, shown below. is a screen capture of the record creation screen.

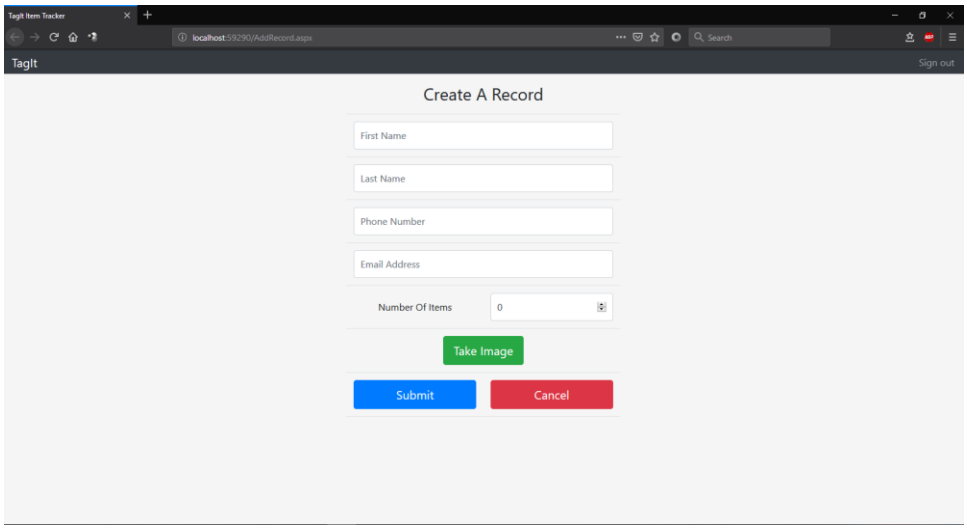


Figure 5; Create Record Desktop

Figure 6; Create Record Mobile, shown below, is a screen capture of the record creation screen.

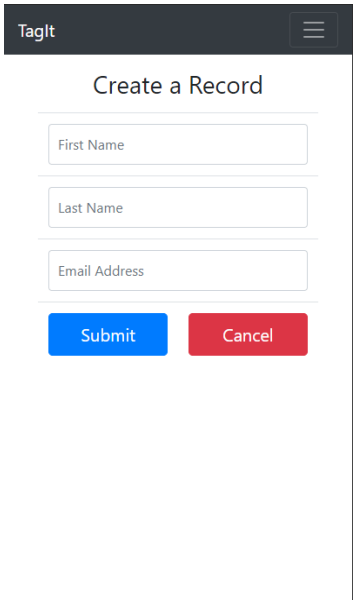


Figure 6; Create Record Mobile

The record creation page is utilized by attendants to enter the required and optional patron details. Once the record has been created, items to be checked can be imaged and moved to the storage location.

4.1.3 Place Item

Figure 7; Place Item Desktop, shown below, is a screen capture of the Place Item screen.

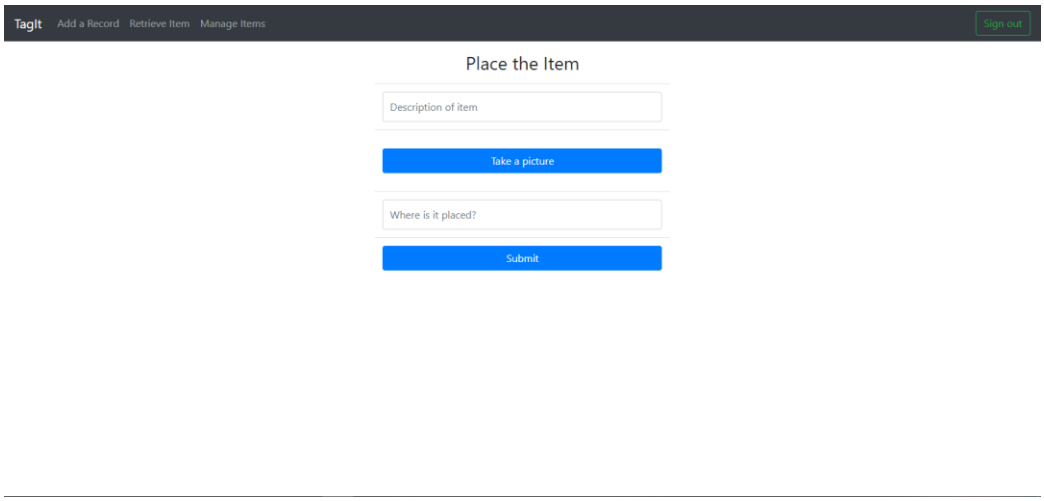


Figure 7; Place Item Desktop

Figure 8; Place Item Mobile, shown below, is a screen capture of the Place Item screen.

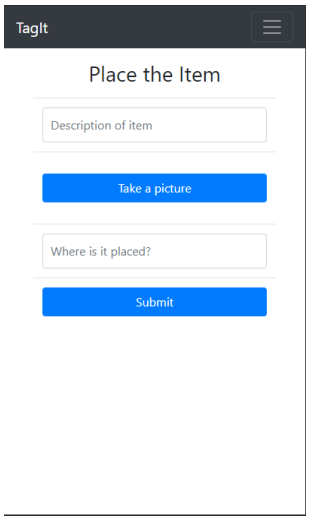


Figure 8; Place Item Mobile

The place item page allows an attendant to assign a description and location to an item in a visit. It also allows the attendant to either attach a photo of the item that has already been taken, or take a current photograph and attach it to the record.

4.1.4 Retrieve Item

Figure 9; Retrieve Item Desktop, shown below, is a screen capture of the Retrieve Item screen.

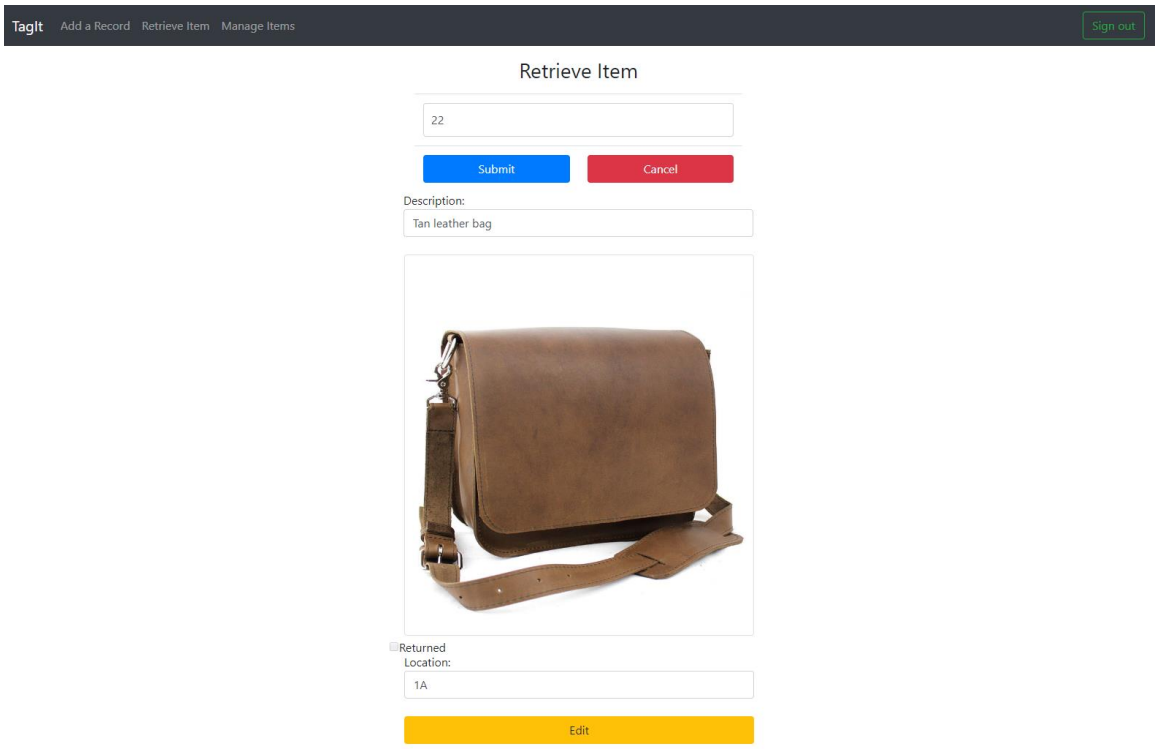


Figure 9; Retrieve Item Desktop

Figure 10; Retrieve Item Mobile, shown below, is a screen capture of the Retrieve Item screen.

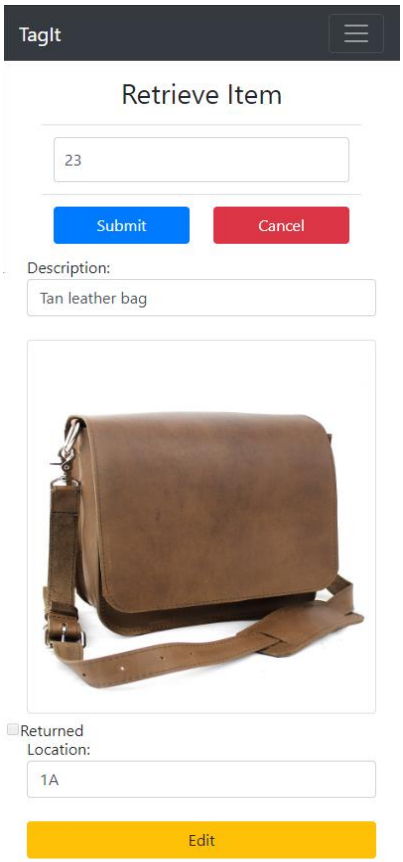


Figure 10; Retrieve Item Mobile

The retrieve item page allows an attendant to return an item to a customer based on the customer's visit ID that they receive in an email. The attendant can also get to this page by scanning the attached QR code that the customer receives in an email.

4.1.5 Manage Items

Figure11; Manage Items Desktop, shown below, is a screen capture of the

Manage Items screen.

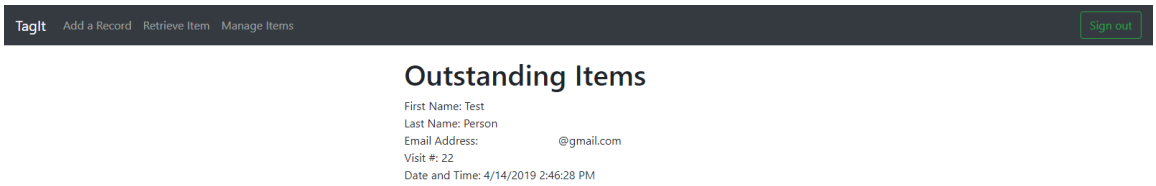


Figure 11; Manage Items Desktop

Figure 12; Manage Items Mobile, shown below, is a screen capture of the Manage Items screen.

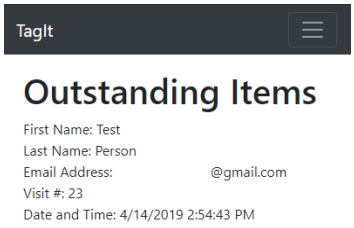


Figure 12; Manage Items Mobile

The manage items page allows an attendant to view all of the outstanding items that haven’t been returned to the rightful customer yet. It provides the first name, last name, email address, visit number, and the date and time that the item was checked.

4.1.6 Network Diagram

Figure 13; Network Diagram, shows the network diagram for the development environment.

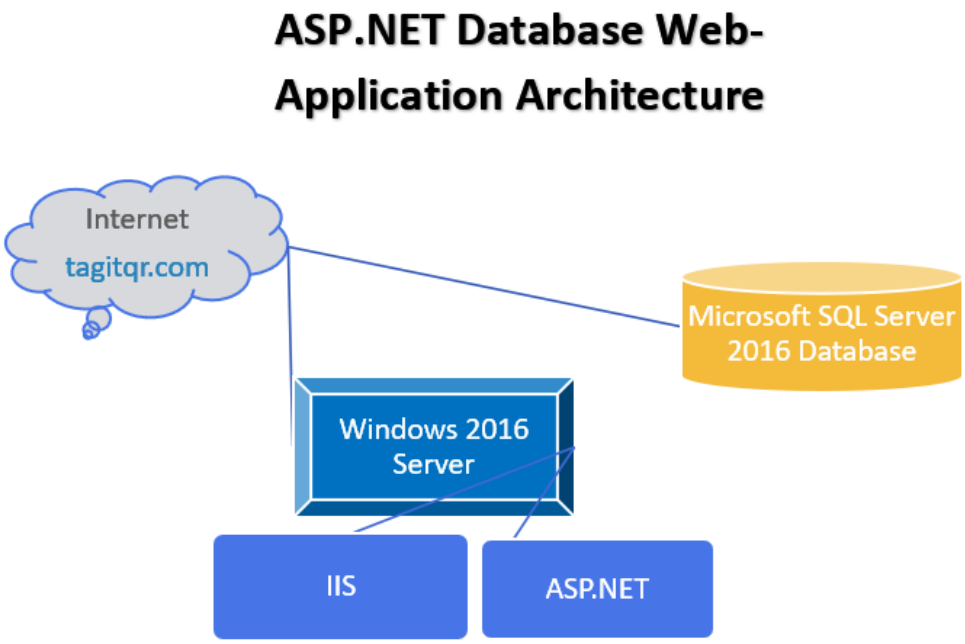


Figure 13; Network Diagram

5. Test Plan

5.1 Overview

This section outlines the Agile ‘test-driven development’ (TDD) approach to testing used for the TagIT Web application. Software development using TDD relies on very short, but repetitive development cycles. The development team coded test cases which define a requirement, improvement, or new function. The goal was to minimize coding time/quantity required to achieve or pass the test, and then refactor that code. Tests were added for each requirement or function, then coded, tested, and refactored until all tests completed successfully.

5.1.2 Implementation and Delivery Approach – Test-Driven Development (TDD)

The traditional "waterfall approach" works well on projects with static requirements and stringent timelines. Large, distributed teams can also benefit from the method. An Agile approach can be great for smaller, more flexible teams, permitting continual code improvements to meet the requirements.

For the TagIT project, Agile’s test-driven approach was chosen as the best option. TDD can help avoid functional complexity and simplify the code base. The approach also ensures each component of the TagIT Web application is meeting the requirements, through the efficient use of repetitive testing.

The development lead was assembled the team’s requirement definitions and quickly outlined the code schema. The use of TDD enabled coding and testing in smaller, more

efficient “chunks”. In this way, code was tested in the early stages of development, simplifying problem resolution, and reducing code complexity within the application. Figure 13; Test-Driven Development, below, is a graphic representation of the TDD method used for the TagIT project.

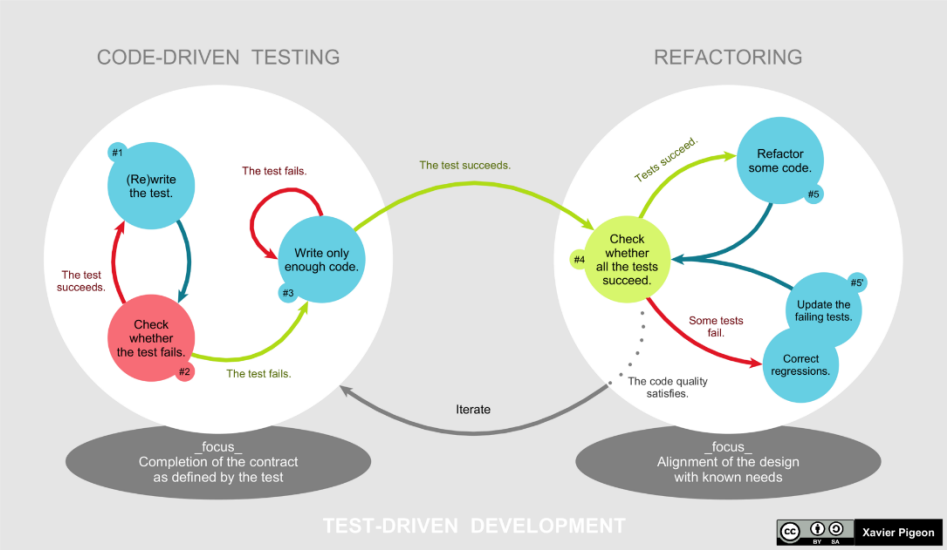


Figure 13; Test-Driven Development

5.1.3 Project Variables

As shown in the following, Figure 14; Project Variables, successful project management requires a clear understanding of the underlying variables of scope, time, resource, and quality. The project scope, or statement of work, outlines the project boundaries and requirements, and defines team member responsibilities. Project scope establishes what constitutes the satisfactory completion of the project and how completion will be measured and verified. Tasks, work schedules, and budget are guided by the scope, and can help the team maintain focus.

Scope has a large impact on a projects time and cost. As scope increases or decreases, so do time and cost. Quality can affect time and cost bilaterally. Poor quality can raise time

and cost through the need to rewrite code and repeat testing. Raising the desired level of quality can have the same effect for the same reasons.

Figure 14; Project Variables

5.2 Test Levels

The development team serve as code-level testers, assisted by other team members as needed. Functional testing includes known bad data types, such as poorly formed phone numbers, email addresses, QR codes, and item image failures.

5.2.1 Operation Criteria

- Browser agnostic
- Broad device compatibility
- Customizable by customer

5.2.2 Test Case

Attendant:

- Login to web-based application
    - Create new record
    - Enter customer phone or email

- Add to new record item image(s) with device
    - Many to one
    - Single or Multiple images
    - Tied to single record
    - Verify item count
- Open and add images to existing record
    - Many to one
    - Single or Multiple images
    - Tied to single record
    - Verify item count
- Stow items in storage location
    - Scan bin QR code
        - Many to many
    - Repeat for each item
        - Verify item count
- Complete record entry
    - Initiate SMS/Email to patron
        - Contains link to record with location and images
- Patron follows link
    - Can see record data (images and location)
    - Can request return of item(s)
        - Any or all
    - Verifies receipt of item(s)
- Record closes when all items marked returned
    - By patron (automatic)
    - By attendant (manual)
    - Record reflects status
- System status
    - Dynamic item count
        - Checked / returned
    - SMS/Email to patron for item not retrieved

5.2.3 Logging and Reporting

Failures and defects (fail condition) will be recorded during testing and reported to the lead developer. The defect will be replicated, if possible, and corrected. The application will then be resubmitted to testing process. These steps will be repeated until the satisfactory resolution (pass condition) of the defect.

5.2.4 Test Report

Table 5; Test Report, below, charts the test results gathered during the development of TagIT.

Req #	Input	Expected Output	Actual Output	Pass / Fail	Notes	Date
1	Customer Name and email	Store Data to Database	N/A	Fail	ID for visit is showing negative.	2/8/2019
1	Customer Name and email	Store Data to Database	N/A	Pass	Subsequent data follows numerical order.	2/8/2019
2	Item Description	Store Data and tie it to user visit	N/A	Pass		2/8/2019
3	Stow Item	Store location in database	N/A	Pass		2/8/2019
4	Send Email with info. to customer	Email with QR code and location		Fail	Email not sending QR code	2/11/2019
5	Enter location to retrieve item	Item and information returned	Item description and location returned	Pass		2/11/2019
6	Item retrieved	Record shows as retrieved	Item shows as retrieved in the database	Pass		2/11/2019

Table 5; Test Report

6. Conclusion

6.1 Senior Design Conclusion

The chief design considerations for the TagIT project were to make the system flexible and hardware agnostic, to replace the paper claim-check ticket, and make using TagIT as transparent as possible for the client. A Web based application offered the greatest range of possibilities. This solution would fulfill the technical requirements of being multiplatform, operating on the greatest range of hardware, and simplifying client operation. As development progressed, we recognized an added benefit of removing most responsibility from the patron/end-user.

Since TagIT is to be hardware agnostic and able to run on mobile platforms, Bootstrap became the obvious choice for the much of design work. Bootstrap would allow us to design our solution like a website, but with more responsive mobile operation. Additional designs could then be accomplished via HTML, CSS, and JavaScript. With ASP.NET as our primary framework, we were able to institute templates with which we could design the entire solution. The templates would allow us to focus on function rather than design. On the server side, C# was chosen as our development language, both for its familiarity and for its powerful capabilities. We used C# to connect to our SQL Server instance and perform the necessary SQL queries. C# was also used to generate our QR codes and send emails. While utilizing query strings, a QR code could be generated that would allow an attendant to scan the code and view that specific client's record. This fulfilled our other requirement of eliminating the paper ticket.

Like C#, we chose a SQL database because of its wide acceptance in the business world and our familiarity in its usage. We believe these factors will lead to decreased costs running and maintaining the solution and simplify TagIT implementations.

Working out the roles and responsibilities turned out to be much easier than anticipated. The team dynamics certainly helped with that. In addition to our weekly class meetings, we met twice a week during the Fall semester changing to once a week for Spring This provided us ample opportunity to brainstorm ideas while uncovering possible problems. The greatest challenge seemed to be how we would make the best use of QR codes to enhance the functionality of the TagIT application. Several hours of on-line research and frequent team discussions led to what we believe will be an optimal application to meet the needs of the product.

George Santen has been organizing the infrastructure tasks, while Aaron Westermeyer developed the SQL schema. Aaron and Craig Slusher are working together on the front-end development. Craig is also taking care of Project Management tasks, secured the TagITQR.com domain, and helped configure AWS.

For the Spring semester, the team originally planned on using UC's new Sandbox Virtual System for TagIT's required infrastructure. This would have allowed the roll-out of Microsoft Windows servers running MS SQL Server as well as IIS for the Web site, E-mail and SMS texting. However, there were delays in our initial contacts with UCIT, and a long wait for external Web access to be configured. We eventually found that Sandbox would not be suitable for the needs of our project and began the search for a replacement hosting platform. In the interim, TagIT was hosted locally to reduce the impact on development while continuing progress on the project.

An account was created with Amazon to explore the wide range of possible solutions offered with AWS. As lead developer, Aaron Westermeyer choose MS Visual Studio with .NET using C# and Bootstrap as the development platforms. Bootstrap was chosen for its cohesive front-end web framework tools. These will provide best case integration with IIS and SQL server for our project.

Quite a bit of research was done on how the QR codes could be used effectively with TagIT. We concluded that having a master QR code for each record, as well as static QR codes for each storage ‘bin’ or location would be an efficient method of tracking items. Each item can be imaged under the master record QR code, with the item/image subsequently tied to the QR code of its ‘bin’. A patron could then easily claim a specific item from the record, or the entire collection. It would also simplify the duties of attendants, leading them directly to the requested item, using the details and images stored in the master record.

Application testing demonstrated the effectiveness of the QR codes distributed in the email sent to a patron, successfully linking patrons, items, and storage locations. The system also maintains a count of unclaimed items and the means to notify the patron with retrieval instructions via e-mail or text message.

While we were initially concerned with how the project would be received by the attendees of IT Expo, the crowd response was very positive. We fielded well thought out questions that demonstrated the interest TagIT had generated. Certainly the best part of IT Expo were the interactions with the public and the possibilities they imagined for the TagIT solution.

Appendix A

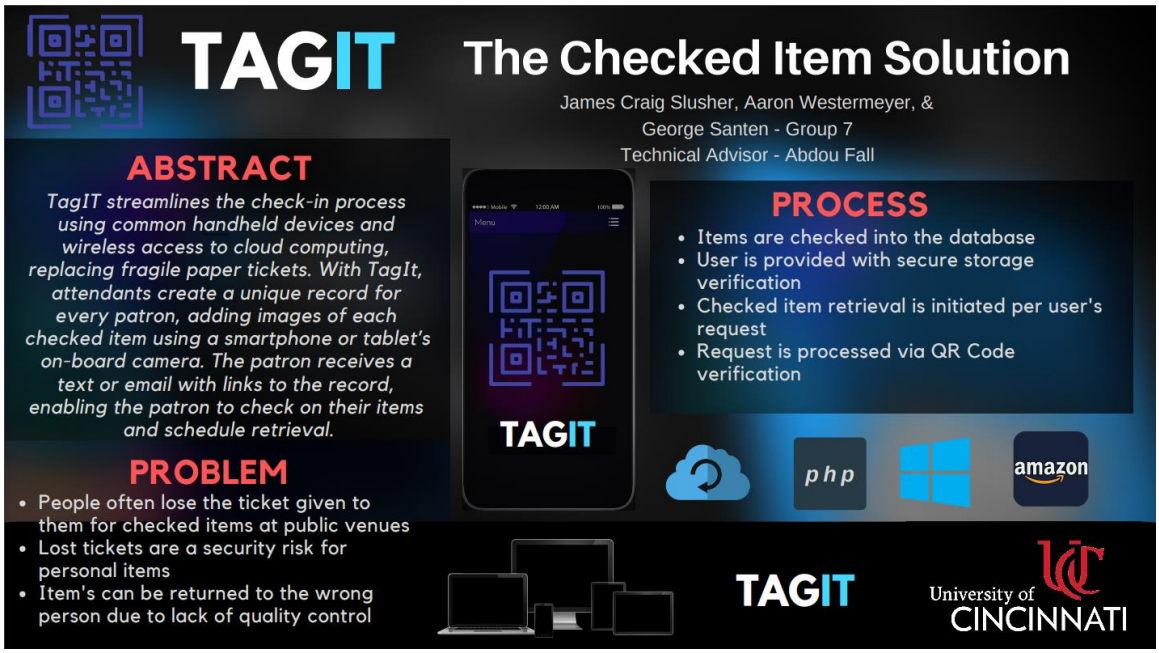


Figure 14; TagIT Project Poster

Appendix B

References

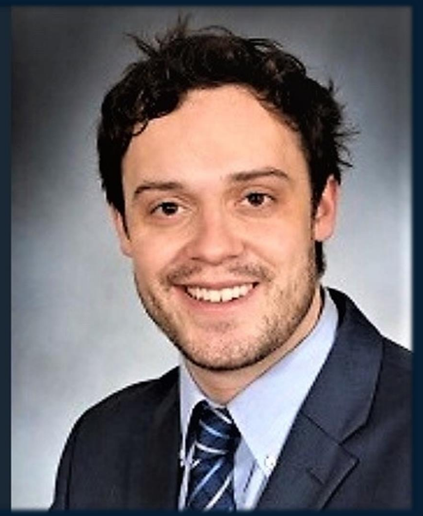
Bruni, F. (2018). *A Ticket to Frustration*. [on-line] Diner's Journal Blog. Available at: https://dinersjournal.blogs.nytimes.com/2009/02/26/a-ticket-to-frustration/ [Accessed 26 Aug. 2018].

The Checked Item Solution

Team 7 Advisor – Abdou Fall

TAGIT

UC Tech Expo 2019



Craig Slusher

- Project Manager
- Development Track

Aaron Westermeyer

- Lead Developer
- Development Track

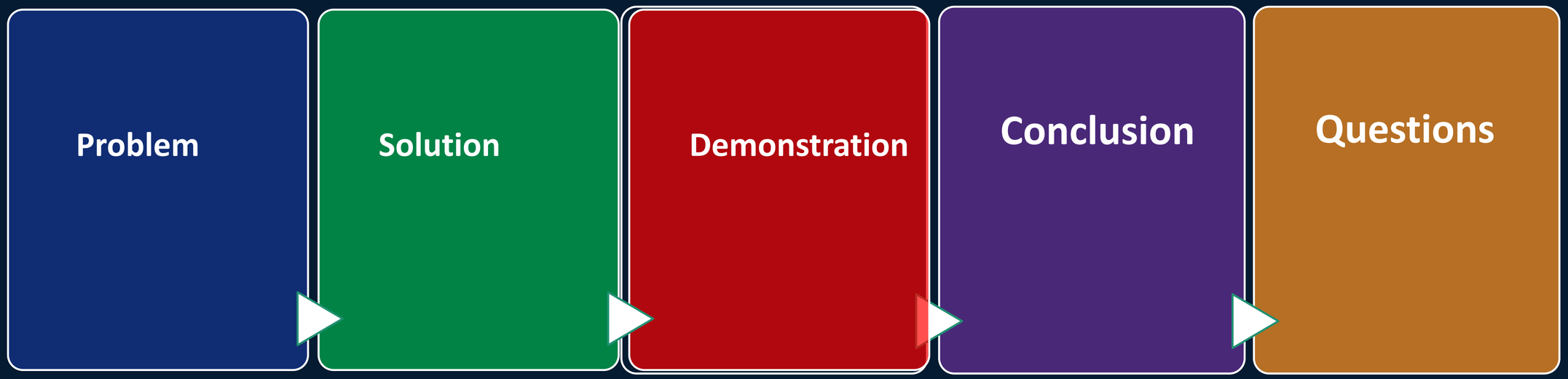
George Santen

- Infrastructure
- Networking / Systems Track

Team 7 Advisor – Abdou Fall

Agenda

...



The Problem

...

A dependable electronic checked item tracking system is not currently available for most organizations

TAGIT

- With TagIT, an attendant will create a unique record for each patron, adding images of their checked items using a smartphone or tablet’s on-board camera

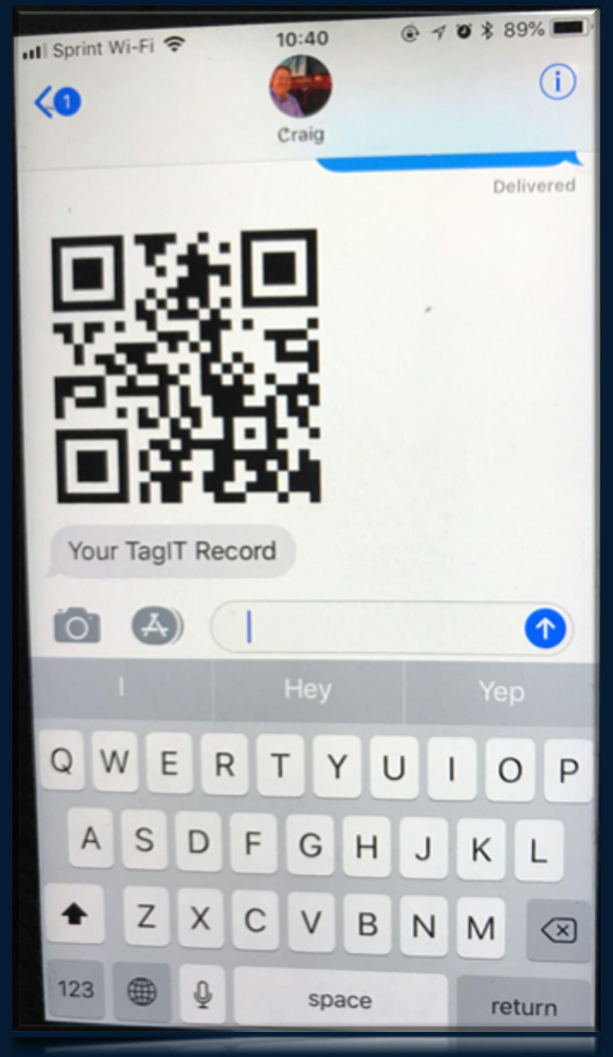
- A unique QR code will be utilized to aid in tracking the patron’s items

TAGIT

TAGIT

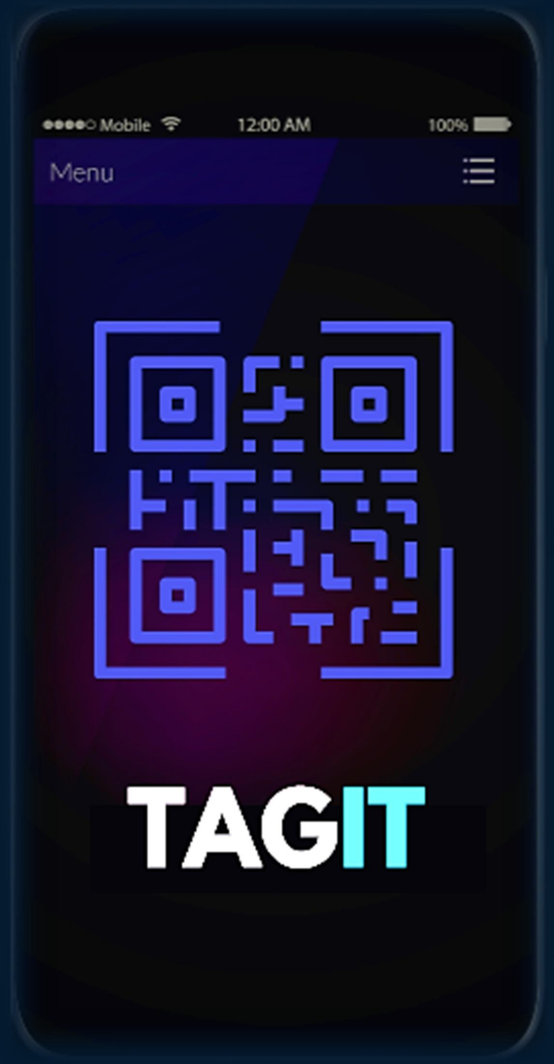
- Once the items have been imaged and added to the record, the patron will receive a text or E-mail with a Web link to the record

- This QR code will be scanned by the attendant for item retrieval



TAGIT

- Runs on industry-standard devices
- Simplifies the user experience
- Ensures the return of items to the correct party
- Provides a process for the return of unclaimed items



Development Environment

- ASP.NET C#
- BOOTSTRAP
- SQL SERVER

TAGIT

DEMO

- https://youtube/9QhE4fnaUIg

Questions?