RFID Tracking System

A thesis submitted to the
Faculty of the Electrical & Computer Engineering Technology Program
of the University of Cincinnati

This Document is partial fulfillment of the
Requirements for the degree of
Bachelor of Science
In Electrical or Computer Engineering Technology
At the College of Engineering & Applied Science

by

Alan Placko
Jimmy Bagnola

Bachelor of Science University of Cincinnati

May 2013

Faculty Advisor: Michael Haas
Dear Professor Michael Haas:

Attached is our final report on the “RFID Equipment Tracking System,” which was requested by the CEAS faculty.

The report covers the entire design process that led to the completion of our project. In this report you will see the problem, solution, credibility, and methodology.

We appreciate your time and effort in helping us throughout the process and we appreciate you reviewing this report. If you have any questions or comments, please contact either one of us.

Sincerely,

______________________________________________________________________________

Jimmy Bagnola
(330)-704-6202
jimmy.bagnola@gmail.com

______________________________________________________________________________

Alan Placko
(513)-235-0770
placko@gmail.com
ACKNOWLEDGEMENTS

We would like to thank Professor Michael Haas for his guidance and patience with this project. His ability to guide us in the right direction was a valuable tool for us. Without his classes that we have previously taken this project wouldn’t have been as successful as it was.
Figures

Figure 1: Tech Expo Display ........................................................................................................ 7
Figure 2: Visual Studio Project Structure ..................................................................................... 10
Figure 3: Application Overview ..................................................................................................... 11
Figure 4: Scanned Item Tab ............................................................................................................ 11
Figure 5: Active RFID .................................................................................................................... 12
Figure 6: Passive RFID from http://www.epc-rfid.info/rfid .......................................................... 13
Figure 7: Database Design ............................................................................................................. 14
Figure 8: Original Concept, Tool Room .......................................................................................... 16
Figure 9: Smart Room, second brainstorm .................................................................................... 17
Figure 10: Tech Expo Table Set up ................................................................................................ 18
Figure 11: Passive RFID Sticker .................................................................................................... 19
Figure 12: Gantt Chart .................................................................................................................... 21
Figure 13: Tech Expo Demonstration ............................................................................................. 23

Tables

Table 1: Initial Budget .................................................................................................................... 20
Table 2: Final Budget ..................................................................................................................... 20
The RFID Lab Equipment Tracking System is a prototype of a system that allows for a large variety of implementations. A Graphical User Interface (GUI) was developed using C# and Windows Presentation Foundation (WPF) for the RFID Tracking System. This GUI allows the users to scan items in and out of the equipment Smart Room, keeps track of important items by making sure that they are being used in authorized areas, and keeps inventory on disposable items. Then, back logs of information are saved on the database in order for data collection, to be evaluated and used to help improve usage of the items being tracked.
INTRODUCTION
The following report outlines the ten-month process involved in conceiving, conceptualizing, designing, prototyping, and completing the RFID Equipment Tracking System.

The RFID Equipment Tracking System allows tracking and analyzing equipment in a work or school environment. The primary purpose is to allow the administration of an institution to keep track and analyze trends in the equipment usage. The ability to know who uses the equipment, when they do is a very valuable thing to know. The system provides a way for the upper management to know when to order items to restock. This instant feedback is a very valuable tool and the RFID Equipment Tracking System provides this.
Problem

Currently there are countless companies, schools, homes, and many other applications that would benefit from equipment tracking. An example of this need could be shown from the theft of two vials containing a virus from Michigan State University’s lab in 2002. Or on a less severe scale, Universities would be able to benefit by tracking their lab equipment’s location and keeping track of what equipment won’t be returned (whether it was a consumable item that needs restocked or an item that was broken and needs to be serviced). This would enable the lab to remain well stocked and in working order, while preventing theft.

Problem Statement: Most labs lack an efficient method of tracking lab equipment and supplies

Solution

We have come up with a solution that addresses these problems, along with other common annoyances associated with keeping track of equipment. The main functionality of the tracking system is to provide real time updates and analysis of the usage of items. Everything in the system is being logged to help keep these live updates available for the upper management.

Credibility

The collective knowledge between Jimmy Bagnola and Alan Placko helped complete the goals set forward in the beginning of this senior design project.

Jimmy Bagnola

Jimmy is qualified to work on this project because of several past experiences. School related, he developed software for a group of Biomedical Engineers last year for their senior project. The project was to create a system to mix fluids to be injected into the human heart. The software he wrote interacted with a relay that was used to power pumps to mix the material. The software allowed the user to keep track of how much of the fluids they were using, and to create mixtures with ease. Also, by taking CET III and GNP, he has learned a lot about programming and communicating with hardware and different programming languages.

Outside of school, he co-oped at both the Timken Company and Toyota Motor Engineering manufacturing North America. At the Timken Company, he developed a gauge that measured the rollers of bearing. It was able to measure with a tolerance of approximately 6 millionths of an inch. This was achieved by writing software to communicate with a linear variable differential transformer and converting the voltage into extremely accurate measurements. At Toyota, Jimmy worked with multiple
programming interfaces to work with PLC’s and to create screens to be displayed on
television throughout the plants.

Lastly, Jimmy has spent a lot of time coordinating events, and acting as a leader in many
different atmospheres. Including: President of the UC bowling team, being a Peer Leader
at the University of Cincinnati, being on the chair of the community service committee at
Toyota and setting up volunteering events, winning the Timken video challenge
(http://www.timken.com/en-us/careers/UniversityRecruiting/Pages/Internships.aspx view
the video on this page) and being a project lead at both Timken and Toyota on multiple
projects. This shows that he is dedicated to producing the best work possible.

Alan Placko

Alan Placko is qualified to work on this project due to his past experience of working on
projects pertaining to the software aspect of this proposal. At Honeywell he wrote
software to test and improve gas line meters. This software was designed to make sure
that the firmware and overall functionality of the meters were working efficiently and
correctly. While working at his second co-op, he also designed and implemented a User
Interface that is used by multiple users spanning a few Public Safety Associations across
the Midwest. He has experience from this project that can be relayed into this project
since a user interface is one of the main components in the preliminary design.

Along with his extensive experience in software at his co-ops, Alan has excelled in a few courses
on campus. Embedded systems was his easiest class- whereas most students barely completed a
few labs. By taking CET III, CET III, and GNP he has also learned how to effectively design
programs. Through all of his programming experience, he is able to learn and adapt to new
standards of coding to implement a solution for the given task.

Goals and Methodology

This project consists of two different parts, hardware and software. The two primary objectives
for the system are to convey information back to the administrator in an updated real-time way,
and to provide information to the users of the equipment when certain items are currently
available. The first objective is met by developing a database for the software to log information
in. The second objective is met by

Overview

The remainder of the final report outlines in detail how the project was completed. This report
includes the following sections: method and materials and discussion, results, budget, timeline,
and future recommendations.
METHODS AND MATERIALS

Software and Applications

The software for this project included three separate projects in Visual Studio. The three solution files included the smart room desktop application. Figure 2 shows the structure of the project. The AccountManagerLibrary project is a C# library that allows membership functionality in both the website and desktop application. There are two other libraries called TrackerBLL (Business Logic Layer) and TrackerDAL (Data Access Layer).

![Visual Studio Project Structure](Figure 2: Visual Studio Project Structure)

The website is built using the .NET framework, more specifically using an MVC pattern over top the Razor engine. This allowed quick prototyping and fast implementation. The TrackerWpf (Windows Presentation Foundation) project is the desktop application.

We wanted the user interface for the desktop application to be easy for the user to use. We streamlined the look and feel to be sleek and clean. There are two primary areas of the application: the content tabs and options tab. Depending on what you are doing in the application will dictate what you see. Figure 3 shows the Account Dashboard tab and its related options tab (on the right). Each tab is allowed to be closed, and re-opened from the menu item (“View > Recently Closed”)
Using different log in credentials, an admin could log in to the system to allow more functionality in the system. When a user scans an item it will display that items tab, however, when an admin scans a new item, they are allowed to enter it into the system. Figure 4 shows what would be displayed when a user scanned an item after they have previously checked the item out.
In Figures 3 and 4 you can see the status bar at the bottom of the application. This served as a quick way to show the user what is actually happening. The different colors correlate to different outcomes. The blue was an idle color serving as a default color. A green color usually meant the user’s action in the system was successful. The red-orange color was displayed when anything went wrong (e.g. scanning an item that doesn’t exist in the system currently). This clear indication allows for the easy usability.

Using the .NET framework allowed quick prototyping and implementation of this project. There were also times that we ran into difficulties when trying to implement more complicated architecture. MVVM (Model, View, View-Model) architecture allows you to keep your data and business logic separate from how your application actually looks. This introduced a lot of headaches when trying to figure out how to cleanly use this type of architecture.

**Hardware**

Active RFID

In order to track sensitive items in our system we use active RFID tags and readers. We chose to do this because it allows us to track and see if these items are being used in authorized areas. We are able to control how large of areas to cover by choosing a proper antenna for the system. By placing readers in multiple labs, or various access points, we are able to track the items and verify that they are being used in authorized areas. For example, if a company has proprietary lab equipment that can only be used in certain areas or a project that is not allowed to leave a secured area. The system is able to check and make sure that the items are being used in authorized areas, when it recognizes that these items are outside of authorized areas for too long, it goes into alarm mode and notifies the administrator immediately. In figure 5, a simple example is given. When the item leaves the “smart room” a timer is started, as long as the equipment gets back into an authorized area before the timer has expired, then the alarm will not go off.

![System recognizes the item is in an approved area](image)

Figure 5: Active RFID
Passive RFID

In order to keep track of the standard items, we are using 125 KHz RFID tags and a reader. Passive RFID has a much shorter range than passive RFID, typically under a foot, but there is no battery and the cost for tags is much cheaper. We chose to do this because a large percentage of companies that would be interested in using our system, already use a RFID tag on their nametag to get into the lab area. This allows us to be able to automatically log users into the system when they unlock the door to get in the lab.

These standard items are items that aren’t terrible important, but it is still good to know who is currently using them. For example, if somebody in a company wants to use a soldering iron, they are able to check it out. Then other members are able to look in the system to see who is using the soldering iron, if they also need to use it.

Figure 6 shows the general set up of any RFID system,

![Passive RFID Diagram](http://www.epc-rfid.info/rfid)

**Database Implementation**

The database was used in the system to keep track of all the items that were currently in the system. The database was used by all three software projects in the systems, which were discussed in the Software and Applications section.
As stated above, the database has to accommodate storing data for the complete system. Database tables were created for keeping track of users, items, and notifications. Subsequently there are other many-to-many relationship tables in the database to help link users to items, or to link RFID tags to specific users or items.

The database also inherently works with the ASP website that was mentioned in the software section of this document. The security measures that are built in with the ASP membership classes worked well when integrating it into the desktop application.

**DISCUSSION**

*Original Concepts*

We started with the idea of an automated toolbox vs. a smart home. We were thinking that we would be able to do one concept or the other. But after a lot of research, we found that there was so much already in existence in home automation that we would have to be more specific in our topic. We discussed several ideas with the automated toolbox, and eventually decided that we should combine the ideas into a smart lab room.
Alternative Applications

The objective of this project is to design and prototype a system that can be used to track items in different contexts by using RFID tracking and multiple access points. This project will narrow its scope to focus on one application, but it should show the flexibility of the system to be used in other applications.

Lab Equipment Tracking:

A potential system with this design proposal could include a way to track lab equipment. This system could be used in either a professional or academic environment. The idea is to allow users to obtain equipment whenever they want, which adds value to this system. The school or company would be able to keep track of the equipment, which would allow them to know who is using certain items and for how long. A remote access user interface could allow the students and employees to know if certain items are available or when they will become available. This will help with the students or employees effectively plan their work.

Tracking Deadly Viruses #2:

A second potential use of this system could be used in a lab that contains deadly viruses. By being able to keep track of when viruses were taken out of the freezer using RFID tags on the petri dishes, and setting timers to alert users when the virus must be returned. The system would be able to track if the viruses are being analyzed in secure locations; and setting off alarms if the virus has been out of these secured areas for too long. This would also enable the user to establish trends and be able to predict when it is most likely time, or person to check out this virus in the future.

Minimum Security Prisons #3:

A third potential use of this system could be used in a minimum security prison by having inmates wear an RFID tag and having multiple access points throughout the prison. By having multiple access points, the prison would be able to track where inmates are and set alarms for certain conditions. For example: if there are 10 inmates in the same room the system could notify the prison security that there are a lot of people in one room, and they could be aware of the condition. The system could also be set to notify guards when an inmate is in a location they shouldn’t be in, for example when the inmate is in the cafeteria but should be in the conference room. The system would be able to add value by increasing the awareness of the prison and being able to predict trends in inmates behavior based off of the data collected.
**Brainstorming**

Figure 8 shows one of our concepts of the smart room, with the main concept being that the whole room is connected. We could use a weight sensor on each of the drawers to keep track of which small tools went, and have a touch screen interface on the storage container for disposable items so that users would be able to easily check items out of the system. In this system, the user would be able to use his or her nametag (passive RFID, barcode reader, magnetic strip or etc.) and then the user would not have to do anything other than leave with the items and the system would be able to know which items the user had.

![Figure 8: Original Concept, Tool Room](image)

When we continued to move forward, we started to come up with questions. What if more than one user is in the room at once? Could we just have a passive RFID reader on the doorway, so that we can just keep track of which item left with what user?
After a lot of research and narrowing down the project, we were thinking that we could still do the smart room; but unless we used ultra-high frequency RFID that we wouldn’t be able to just read any item that goes through the doorway. We decided that we would just use a passive RFID reader, and have the user manually check out each item. We were also thinking that we would like the room to have isolated areas that would be able to be unlocked or locked based on the user’s access level. Figure 9 shows our concept of having different sections of the smart room for different purposes.

Figure 9: Smart Room, Second Brainstorm
At the Tech Expo

Figure 10: Tech Expo Table Set up

**Sensitive Items**

Active RFID tags were used to track the sensitive items. These tags were displayed in the bottom right side of the table that we presented at the tech expo, as seen in figure 10. We put one of these tags into an enclosure with an on/off switch. By turning the tag off, we were able to simulate the item leaving the authorized area, and thus notifying the manager. Unfortunately we were unable to demonstrate having multiple access points due to the size of our table, but we were able to demonstrate when an item goes missing.

**Standard Items**

Passive RFID tags were used to track the standard items, by using RFID stickers; we were able to mark all of the standard items. These are all of the items located in front of the poster in Figure 10. The system allowed users to log in using a passive RFID “name tag,” then tracked
who checked out items to demonstrate a user checking items in and out of the system. Figure 11 shows one of the RFID stickers that we used to mark the standard items; these stickers have a very strong adhesive on them.

![Passive RFID Sticker](image)

**Figure 11: Passive RFID Sticker**

**Disposable Items**

To demonstrate disposable items at the Tech Expo, we filled a tackle box with items we considered to be disposable. Such as: pens, pencils, dry erase markers, screws and etc. We used these items to show that by using the software, we were easily able to check out the number of items that we were removing from the system.

**Web Application**

The web application did not have full functionality at the tech expo, but we were able to revert to an older version in order to show that we were able to demonstrate the notifications as well as being able to view if items are currently checked out or not.

**Budget**

The initial budget for this project included all the necessary components to build a working system. In Table 1, which was created in the first month of the project, all the parts we thought would be needed in our project were included.
We were able to reduce the cost of the overall project by narrowing down our scope of work. Eliminating the touch screen monitor, webcam, and motion sensor from our initial budget helped make our budget smaller. We also were able to make the budget lower by removing the computer and monitor out of the budget. This was under the assumption that there was already a computer, or access to a computer to be used in the system. Table 2 shows our final budget, the prices for the Active RFID Readers and tags are approximate, since it would be an independent per any installation’s specifications.

<table>
<thead>
<tr>
<th>Equipment Tracker Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
</tr>
<tr>
<td>Passive RFID Tag reader</td>
</tr>
<tr>
<td>Passive RFID Tags</td>
</tr>
<tr>
<td>Personal Computer</td>
</tr>
<tr>
<td>Monitor</td>
</tr>
<tr>
<td>Touch Screen Monitor</td>
</tr>
<tr>
<td>Motion Sensor</td>
</tr>
<tr>
<td>Webcam</td>
</tr>
<tr>
<td>Active RFID tag</td>
</tr>
<tr>
<td><strong>Total Item Price</strong>:</td>
</tr>
</tbody>
</table>

| Work Hours | 250 Hours | Price per hour | $40.00 | **Total Labor Price**: | $10,000.00 |

Table 1: Initial Budget

<table>
<thead>
<tr>
<th>Component</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labor (2 people, $20 per hour for 100 hours)</td>
<td>$4000.00</td>
</tr>
<tr>
<td>Passive RFID Tag and reader kit</td>
<td>$45.57</td>
</tr>
<tr>
<td>Extra 125Khz Tags</td>
<td>$11.87</td>
</tr>
<tr>
<td>Active RFID Readers</td>
<td>$500.00</td>
</tr>
<tr>
<td>Active RFID tags</td>
<td>$100.00</td>
</tr>
<tr>
<td>Visual Studio</td>
<td>free</td>
</tr>
<tr>
<td>SQL Server Management</td>
<td>free</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$4657.44</td>
</tr>
</tbody>
</table>

Table 2: Final Budget
Timeline

A timeline was created the first semester of work for the senior design project. The timeline helped create a schedule that allowed timely completion of the project. Figure 9 shows the gantt chart that was created for the timeline. This chart helped both members of the team stay on task. Time management was crucial in the new semester conversion, the Tech Expo was pushed closer than it normally has been in the past.

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Lists of Requirements</td>
<td>20 days</td>
<td>Wed 9/18/12</td>
<td>Tue 10/23/12</td>
</tr>
<tr>
<td>Create detailed Gantt Chart</td>
<td>14 days</td>
<td>Wed 10/3/12</td>
<td>Mon 10/22/12</td>
</tr>
<tr>
<td>Research Technologies in field</td>
<td>20 days</td>
<td>Wed 10/3/12</td>
<td>Tue 10/30/12</td>
</tr>
<tr>
<td>Create Application Design Specification Document</td>
<td>14 days</td>
<td>Thu 10/11/12</td>
<td>Tue 10/30/12</td>
</tr>
<tr>
<td>Create database design specification</td>
<td>7 days</td>
<td>Wed 10/24/12</td>
<td>Thu 11/1/12</td>
</tr>
<tr>
<td>Develop Active RFID portion of System</td>
<td>12 days</td>
<td>Wed 10/24/12</td>
<td>Thu 11/22/12</td>
</tr>
<tr>
<td>Create comparison document for hardware to use</td>
<td>7 days</td>
<td>Tue 10/30/12</td>
<td>Wed 11/7/12</td>
</tr>
<tr>
<td>Purchase or locate necessary hardware</td>
<td>14 days</td>
<td>Wed 11/7/12</td>
<td>Mon 11/26/12</td>
</tr>
<tr>
<td>Implement hardware into application</td>
<td>14 days</td>
<td>Wed 11/21/12</td>
<td>Thu 12/18/13</td>
</tr>
<tr>
<td>Develop Passive RFID portion of System</td>
<td>30 days</td>
<td>Thu 11/22/12</td>
<td>Wed 1/2/13</td>
</tr>
<tr>
<td>Integrate sections of System</td>
<td>10 days</td>
<td>Sun 1/3/13</td>
<td>Thu 1/24/13</td>
</tr>
<tr>
<td>Create demonstration system</td>
<td>27 days</td>
<td>Thu 1/24/13</td>
<td>Fri 3/1/13</td>
</tr>
<tr>
<td>Tweak programs to make an entertaining demo</td>
<td>15 days</td>
<td>Fri 3/1/13</td>
<td>Thu 3/11/13</td>
</tr>
<tr>
<td>Create Presentation for Tech Expo</td>
<td>7 days</td>
<td>Fri 3/15/13</td>
<td>Mon 3/18/13</td>
</tr>
<tr>
<td>Tech Expo</td>
<td>1 day</td>
<td>Tue 3/26/13</td>
<td>Tue 3/26/13</td>
</tr>
</tbody>
</table>

Figure 12: Gantt Chart
**Future Possibilities**

Like any project, there are endless possibilities that could be added on to the system. Here is the list of possibilities that we created for our poster at the Tech expo.

- Track trends of disposable items to improve replenishing stock
- Reserve items in the system
- Replace the need for a computer by using an Ethernet access point
- Store picture of the items before and after use, and store them in a database
- Lock and unlock doors with a lab depending on the user’s access level
- Phone and tablet applications
- Advanced security measures, such as cameras and motion detectors
- Text message/email notifications
CONCLUSION

The result of this project was a functioning prototype of our system. This system tracks items with three different tiers. Sensitive items that are tracked using active RFID tags, these items are also able to be tracked to ensure that they are being used in authorized areas. It also kept track of standard items by having users check items in and out using passive RFID tags and readers. For disposable items, it keeps an updated inventory of items taken out of the system, complete with a notification system to alert the admin that there is a low inventory. Lastly, the app application allows users to check and see if items are currently available as well as giving the admin the ability to see any recent activity in the system.

Figure 13: Tech Expo Demonstration
REFERENCES

APPENDIX A

Source Code

```csharp
namespace TrackerWeb.Models
{
    // models for the ASP.net website to use to insert objects in the MVC pattern

    public class ChangePasswordModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Current password")]
        public string OldPassword { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The \{0\} must be at least \{2\} characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        public string ConfirmPassword { get; set; }
    }

    public class LogOnModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
    }

    public class RegisterModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.EmailAddress)]
        [Display(Name = "Email address")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The \{0\} must be at least \{2\} characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        public string confirmPassword { get; set; }
    }
}
```
namespace TrackerWeb.Models
{

/// <summary>
/// A class to send information or errors to user inside an alert/message box
/// </summary>

public static class Alerts
{
    #region Types of Alerts
    public const string SUCCESS = "success";
    public const string ATTENTION = "attention";
    public const string ERROR = "error";
    public const string INFORMATION = "info";
    #endregion

    // returns all alert types
    public static string[] ALL
    {
        get { return new[] { SUCCESS, ATTENTION, INFORMATION, ERROR }; } } 

}
}

namespace TrackerWeb.Filters
{

/// <summary>
/// Clears all notifications in the database when an ActionResult method
/// has completely executed.
/// </summary>

public class NotificationClearAttribute : ActionFilterAttribute, IActionFilter
{
    public override void OnResultExecuted(ResultExecutedContext filterContext)
```csharp
var notificationManager = new TrackerService().NotificationManager;
notificationManager.MarkNotifications();
base.OnResultExecuted(filterContext);
}

namespace TrackerWeb.Controllers
{
    public class AccountController : BaseController
    {

        #region [ logon ]

        //
        // GET: /Account/LogOn

        public ActionResult LogOn()
        {
            return View();
        }

        //
        // POST: /Account/LogOn

        [HttpPost]
        public ActionResult LogOn(LogOnModel model, string returnUrl)
        {
            if (ModelState.IsValid)
            {
                if (Membership.ValidateUser(model.UserName, model.Password))
                {
```
FormsAuthentication.SetAuthCookie(model.UserName, model.RememberMe);

if (Url.IsLocalUrl(returnUrl) && returnUrl.Length > 1 && returnUrl.StartsWith("/")
    && !returnUrl.StartsWith("//") && !returnUrl.StartsWith("\\"))
{
    return Redirect(returnUrl);
}
else
{
    return RedirectToAction("Index", "Home");
}
}

// If we got this far, something failed, redisplay form
return View(model);

//
// GET: /Account/LogOff

public ActionResult LogOff()
{
    FormsAuthentication.SignOut();

    return RedirectToAction("Index", "Home");
}
#endregion

#region [ register / change password]

//
// GET: /Account/Register

[Authorize(Roles="admin")]
public ActionResult Register()
{
    return View();
}

//
// POST: /Account/Register

[HttpPost]
public ActionResult Register(RegisterModel model)
{
    if (ModelState.IsValid)
    {
        // Attempt to register the user
        MembershipCreateStatus createStatus;
        Membership.CreateUser(model.UserName, model.Password, model.Email, null, null, true, null, out createStatus);

        if (createStatus == MembershipCreateStatus.Success)
        {
            SendNotification("User Created",
            db.Users.First(user => user.UserName == model.UserName));
            return RedirectToAction("Index", "Users");
        }
    }
}
#endregion
ModelState.AddModelError("", ErrorCodeToString(createStatus));

// If we got this far, something failed, redisplay form
return View(model);
}

// GET: /Account/ChangePassword

[Authorize]
public ActionResult ChangePassword()
{
    return View();
}

// POST: /Account/ChangePassword

[Authorize]
[HttpPost]
public ActionResult ChangePassword(ChangePasswordModel model)
{
    if (ModelState.IsValid)
    {
        // ChangePassword will throw an exception rather
        // than return false in certain failure scenarios.
        bool changePasswordSucceeded;
try {
    MembershipUser currentUser = Membership.GetUser(User.Identity.Name, true /* userIsOnline */);
    changePasswordSucceeded = currentUser.ChangePassword(model.OldPassword, model.NewPassword);
}
catch (Exception)
{
    changePasswordSucceeded = false;
}

if (changePasswordSucceeded)
{
    return RedirectToAction("ChangePasswordSuccess");
}
else
{
    ModelState.AddModelError("", "The current password is incorrect or the new password is invalid.");
}

// If we got this far, something failed, redisplay form
return View(model);

//
// GET: /Account/ChangePasswordSuccess

public ActionResult ChangePasswordSuccess()
{

return View();
}

#endregion

#region Status Codes

private static string ErrorCodeToString(MembershipCreateStatus createStatus)
{
    // See http://go.microsoft.com/fwlink/?LinkID=177550 for
    // a full list of status codes.
    switch (createStatus)
    {
    case MembershipCreateStatus.DuplicateUserName:
        return "User name already exists. Please enter a different user name."
    case MembershipCreateStatus.DuplicateEmail:
        return "A user name for that e-mail address already exists. Please enter a different e-mail address."
    case MembershipCreateStatus.InvalidPassword:
        return "The password provided is invalid. Please enter a valid password value."
    case MembershipCreateStatus.InvalidEmail:
        return "The e-mail address provided is invalid. Please check the value and try again."
    case MembershipCreateStatus.InvalidAnswer:
        return "The password retrieval answer provided is invalid. Please check the value and try again."
    case MembershipCreateStatus.InvalidQuestion:

```
namespace TrackerWeb.Controllers
{

    [NotificationClear]
    }

public class NotificationsController : BaseController
{

    //
    // GET: /Notifications/
    // returns all

```
public ActionResult Index()
{
    var orderedNotifications = Service.NotificationManager.GetAllNotifications();

    return View(orderedNotifications);
    // send all new notifications as the model to the view
}

//
// GET: /Notifications/new

[NotificationClear]
public ActionResult New()
{
    var newNotifications = Service.NotificationManager.GetNewNotifications();

    return View(newNotifications);
    // Show only "new" notifications on the view
}

namespace TrackerWeb.Controllers
{
    [Authorize(Roles="Admin,Manager")]
    public class UsersController : BaseController
    {
        //
        // GET: /Users/
        //
    }
}
public ActionResult Index()
{
    return View(db.Users);
}

//@
//@ GET: /Users/View/5
public ActionResult View(Guid? id)
{
    if (id == null)
    {
        return View(db.Users);
    }
    var user = db.Users.Find(id);
    if (user == null)
    {
        Error("Invalid user id");
        return RedirectToAction("Index");
    }
    return View(user);
}

//@
//@ GET: /Users/Edit/5

public ActionResult Edit(Guid id)
{
    var user = db.Users.Find(id);

if (user == null)
{
    Error("Invalid user id");
}
return View(user);
}

//
// GET: /Users/Edit/Model

[AcceptVerbs(HttpVerbs.Post)]
[HttpPost]
public ActionResult Edit(Users model)
{
    if (ModelState.IsValid)
    {
        var user = db.Users.First(u => u.UserId == model.UserId);
        user.UserName = model.UserName;
        db.SaveChanges();
        Information("User Information Saved successfully");
        return RedirectToAction("index");
    }
    return View(model);
}

public ActionResult Delete(Guid id)
{
    Users user = db.Users.Find(id);
    var user = db.Users.First(u => u.UserId == model.UserId);
    user.UserName = model.UserName;
    db.SaveChanges();
    Information("User Information Saved successfully");
    return RedirectToAction("index");
    return View(model);
}
return View(user);
}

[HttpPost]
public ActionResult Delete(Guid id, FormCollection formData)
{
    Users user = db.Users.FirstOrDefault(u => u.UserId == id);
    try
    {
        user.Memberships.IsLockedOut = true;
        db.Entry(user).State = EntityState.Modified;
        db.SaveChanges();
        Information(string.Format("User \"{0}\" was removed.", user.UserName));
        SendNotification("User Marked as Locked", user);
        return RedirectToAction("Index");
    }
    catch (Exception ex)
    {
        Error(ex.Message);
        return View(user);
    }
}
}

namespace TrackerWpf.Service
{
    public static class RFIDService
{ 

    #region PassiveBoard

    public static SerialPort PassivePort; 

    public static void LoadPassivRFIDPort(string portName)
    {
        try
        {
            PassivePort = new SerialPort();
            PassivePort.BaudRate = 9600;
            PassivePort.PortName = portName;
            PassivePort.Parity = Parity.None;
            PassivePort.DataBits = 8;
            PassivePort.StopBits = StopBits.One;
            PassivePort.Handshake = Handshake.None;
            PassivePort.ReadTimeout = 3000;
            PassivePort.DtrEnable = true;
            PassivePort.ReceivedBytesThreshold = 5;
            // data recieved events now setup inside specific ViewModels
            //port.DataReceived += new SerialDataReceivedEventHandler(port_DataReceived);
            PassivePort.Open();
        }
        catch (Exception ex)
        {
            // TODO: publish No RFID reader event
            MessageBox.Show(ex.Message);
        }
    }

    public static string ReadPassive()
{ return PassivePort.ReadExisting(); }

/// <summary>
/// Waits a little bit so it doesn’t continue to read RFID tags
/// Clears the data and resets the DTR on the port to enabled
/// </summary>
/// <param name="closePort">will close the port if you specify to do so</param>

public static void ClearPassiveData(bool closePort = false)
{
    data = "";
    PassivePort.DiscardInBuffer();
    PassivePort.DtrEnable = true;
    if (closePort) PassivePort.Close();
}

#endregion

#region Parallax

public static SerialPort ParalaxPort;
public static string data = "";

/// <summary>
/// loads the serial port for the Paralax RFID usb reader
/// </summary>

public static void LoadParalaxSerialPort(string portName)
{
    try
    {

ParalaxPort = new SerialPort();
ParalaxPort.BaudRate = 2400;
ParalaxPort.PortName = portName;
ParalaxPort.Parity = Parity.None;
ParalaxPort.DataBits = 8;
ParalaxPort.StopBits = StopBits.One;
ParalaxPort.Handshake = Handshake.None;
ParalaxPort.ReadTimeout = 3000;
ParalaxPort.DtrEnable = true;
ParalaxPort.ReceivedBytesThreshold = 12;
// data recieved events now setup inside specific ViewModels
//port.DataReceived += new SerialDataReceivedEventHandler(port_DataReceived);
ParalaxPort.Open();
}

catch (Exception ex)
{
    // TODO: publish No RFID reader event
    MessageBox.Show(ex.Message);
}

/// <summary>
/// temporarily disables the port and reads the data
/// to reset the port you must call RFIDService.ClearData()
/// </summary>
/// <returns>the RFID value from the port</returns>
public static string ReadParalax()
{
    ParalaxPort.DtrEnable = false;
    return ParalaxPort.ReadExisting();
/// <summary>
/// Waits a little bit so it doesn't continue to read RFID tags
/// Clears the data and resets the DTR on the port to enabled
/// </summary>
/// <param name="closePort">will close the port if you specify to do so</param>

public static void ClearParalaxData(bool closePort = false)
{
    Thread.Sleep(2000); // sleep so the rfid reader doesn't read another rfid tag
    data = "";
    ParalaxPort.DiscardInBuffer();
    ParalaxPort.DtrEnable = true;
    if (closePort) ParalaxPort.Close();
}

namespace AccountManager
{
    using System;
    using System.Collections.Generic;

    public interface IRoleManager
    {
        void AddUserToRole(string userName, string roleName);
        void CreateRole(string roleName);
        void DeleteRole(string roleName);
        ICollection<string> LoadAllRoles();
    }
}
namespace AccountManager
{
    using System;
    using System.Collections.Generic;
    using AccountManager.Entity;

    public interface IUserManager
    {
        #region Create

        User CreateUser(string userName, string password, string email);

        #endregion

        #region Read

        ICollection<User> FindUsers(string userNameOrEmail, int pageIndex, int pageSize, out int totalRecords);
        ICollection<User> FindUsersByEmail(string email, int pageIndex, int pageSize, out int totalRecords);
        ICollection<User> FindUsersByUserName(string userName, int pageIndex, int pageSize, out int totalRecords);
        ICollection<User> LoadAllUsers();
        ICollection<User> LoadAllUsers(int pageIndex, int pageSize, out int totalRecords);

        #endregion
    }
}
User LoadUserById(Guid guid);
User LoadUserByUsername(string userName);

#region

#endregion

#region Update

void UnlockUser(string userName);
void UpdateUser(User user);
void UpdateUserPassword(string userName, string newPassword);

#endregion

#region Delete

void DeleteUser(string userName);

#endregion

namespace TrackerBLL
{

    /// <summary>
    /// a class that allows CRUD operations on the database
    /// </summary>
    public class NotificationManager : INotificationManager
    {
    }
// database entity from the data access layer
private TrackerEntities db { get; set; }

public NotificationManager(TrackerEntities context)
{
    this.db = context;
}

/// <summary>
/// returns the count of how many new notifications there are
/// </summary>
public int NewCount()
{
    return GetNewNotifications().Count;
}

/// <summary>
/// <returns>List of New Notifications</returns>
public ICollection<Notifications> GetNewNotifications()
{
    // return notifications that haven't been viewed descending based on the time notification was created
    return db.Notifications.Where(n => !n.WasViewed)
        .OrderByDescending(n => n.time)
        .ToList();
}
/// <summary>
/// Returns A Collection of all the current notifications on the database
/// **TODO: pagination to only return a certain amount of notifications so it's not as network intensive
/// </summary>
/// <returns>
public ICollection<Notifications> GetAllNotifications()
{
    return db.Notifications.OrderByDescending(n => n.time)
        .ToList();
}

/// <summary>
/// adds a notification that was initialized by a user's action
/// normally user creation, or user account information changed
/// </summary>
/// <param name="user"></param>
/// <param name="message"></param>
public void AddNotification(Users user, string message)
{
    Notifications note = new Notifications();
    note.Message = message;
    note.time = DateTime.Now;
    note.AssociatedUser = user.UserId;
    note.NotificationId = Guid.NewGuid();
    db.Notifications.Add(note);
db.SaveChanges();

/// <summary>
/// add a notification based on a specific item in the system.
/// **TODO** needs to update based on the user making those actions on the item
/// </summary>

/// <param name="item"></param>
/// <param name="message"></param>

public void AddNotification(Items item, Users user, string message)
{
    throw new NotImplementedException();
}

/// <summary>
/// Marks the notifications in the database as "wasviewed"
/// </summary>

/// <returns>True if all the notifications were marked as viewed</returns>

public bool MarkNotifications()
{
    try
    {
        var newNotes = db.Notifications.Where(n => !n.WasViewed);
        foreach (var notification in newNotes)
        {
            notification.WasViewed = true;
        }
    }
}
db.SaveChanges();

return true;
}

catch(Exception)
{
    return false;
}

namespace TrackerBLL
{

    public class UserManager
    {
        // database entity from the data access layer
        private TrackerEntities db { get; set; }

        public UserManager() : this(new TrackerEntities()) { }

        public UserManager(TrackerEntities context)
        {
            this.db = context;
        }

        public List<ItemsByUsers> GetItemCheckout(string username)
        {
            return db.ItemsByUsers.Where(u => u.Users.UserName == username).ToList();
        }
    }
}
public Users GetUserByRFID(string rfid)
{
    return db.Users.FirstOrDefault<Users>(x => x.Memberships.Tags.TagRFID == rfid);
}

public Users GetUserByName(string username)
{
    return db.Users.First<Users>(x => x.UserName.Equals(username));
}

public bool ValidateUser(string username, string password)
{
    var user = this.GetUserByName(username);

    if (user == null) return false;
    else
    {
        var hash = HashPassword(user.Memberships.PasswordSalt, password);
        if (user.Memberships.Password == hash)
            return true;
        else
            return false;
    }
}

#region Hash
private string HashPassword(string Salt, string Password)
{
    Rfc2898DeriveBytes Hasher = new Rfc2898DeriveBytes(Password,
        System.Text.Encoding.Default.GetBytes(Salt), 10000);
    return Convert.ToBase64String(Hasher.GetBytes(25));
}

#endregion

public IEnumerable<Users> GetUsers()
{
    return db.Users;
}

public void UpdateUserRFID(Users user, string rfid)
{
    var i = db.Users.First(u => u.UserName == user.UserName);
    i.Memberships.Tags.TagRFID = rfid;
    this.db.SaveChanges();
}

public void AddDefaultUser(Tags tag)
{
    var guid = Guid.NewGuid();

    Users user = new Users
    {
        
}
UserId = Guid.NewGuid(),
UserName = String.Format("NewUser_{0}", tag.TagRFID),

};
Memberships info = new Memberships()
{
    CreateDate = DateTime.Now,
    UserId = user.UserId,
    RFID = tag.TagID
};
db.Users.Add(user);
db.Memberships.Add(info);
db.SaveChanges();
}
APPENDIX B:

Datasheets

Antenna Circuit Design for RFID Applications

**REVIEW OF A BASIC THEORY FOR RFID ANTENNA DESIGN**

**Current and Magnetic Fields**

Amperes law states that current flowing in a conductor produces a magnetic field around the conductor. The magnetic field produced by a current element, as shown in Figure 1, on a round conductor (wire) with a finite length is given by:

\[
B_\phi = \frac{\mu_0 I}{2\pi r} (\cos \alpha _2 - \cos \alpha _1) \quad \text{(Weber/m}^2\text{)}
\]

where:
- \( I \) = current
- \( r \) = distance from the center of wire
- \( \mu_0 \) = permeability of free space and given as \( 4\pi \times 10^{-7} \) (Henry/meter)

In a special case with an infinitely long wire where:
- \( \alpha _1 = -180^\circ \)
- \( \alpha _2 = 0^\circ \)

Equation 1 can be rewritten as:

\[
B_\phi = \frac{\mu_0 I}{2\pi r} \quad \text{(Weber/m}^2\text{)}
\]

**FIGURE 1: CALCULATION OF MAGNETIC FIELD B AT LOCATION P DUE TO CURRENT I ON A STRAIGHT CONDUCTING WIRE**

This section is written for RF coil designers and RFID system engineers. It reviews basic electromagnetic theories on antenna coils, a procedure for coil design, calculation and measurement of inductance, an antenna tuning method, and read range in RFID applications.
The magnetic field produced by a circular loop antenna is given by:

**EQUATION 3:**

\[
B_z = \frac{\mu_0 I N a^2}{2(a^2 + r^2)^{3/2}}
\]

\[
= \frac{\mu_0 I N a^2}{2} \left( \frac{1}{r^3} \right) \quad \text{for} \quad r^2 >> a^2
\]

where

- \( I \) = current
- \( a \) = radius of loop
- \( r \) = distance from the center of loop
- \( \mu_0 \) = permeability of free space and given as \( 4\pi \times 10^{-7} \) (Henry/meter)

The above equation indicates that the magnetic field strength decays with \( 1/r^3 \). A graphical demonstration is shown in Figure 3. It has maximum amplitude in the plane of the loop and directly proportional to both the current and the number of turns, \( N \).

Equation 3 is often used to calculate the ampere-turn requirement for read range. A few examples that calculate the ampere-turns and the field intensity necessary to power the tag will be given in the following sections.
INDUCED VOLTAGE IN AN ANTENNA COIL

Faraday’s law states that a time-varying magnetic field through a surface bounded by a closed path induces a voltage around the loop.

Figure 4 shows a simple geometry of an RFID application. When the tag and reader antennas are in close proximity, the time-varying magnetic field $\mathbf{B}$ that is produced by a reader antenna coil induces a voltage (called electromotive force or simply EMF) in the closed tag antenna coil. The induced voltage in the coil causes a flow of current on the coil. This is called Faraday’s law. The induced voltage on the tag antenna coil is equal to the time rate of change of the magnetic flux $\Psi$.

**EQUATION 4:**

\[ V = -N \frac{d\Psi}{dt} \]

where:

- $N$ = number of turns in the antenna coil
- $\Psi$ = magnetic flux through each turn

The negative sign shows that the induced voltage acts in such a way as to oppose the magnetic flux producing it. This is known as Lenz’s law and it emphasizes the fact that the direction of current flow in the circuit is such that the induced magnetic field produced by the induced current will oppose the original magnetic field.

The magnetic flux $\Psi$ in Equation 4 is the total magnetic field $\mathbf{B}$ that is passing through the entire surface of the antenna coil, and found by:

**EQUATION 5:**

\[ \Psi = \int \mathbf{B} \cdot d\mathbf{S} \]

where:

- $\mathbf{B}$ = magnetic field given in Equation 2
- $\mathbf{S}$ = surface area of the coil
- $\mathbf{\cdot}$ = inner product (cosine angle between two vectors) of vectors $\mathbf{B}$ and surface area $\mathbf{S}$

Note: Both magnetic field $\mathbf{B}$ and surface $\mathbf{S}$ are vector quantities.

The presentation of inner product of two vectors in Equation 5 suggests that the total magnetic flux $\Psi$ that is passing through the antenna coil is affected by an orientation of the antenna coils. The inner product of two vectors becomes minimized when the cosine angle between the two are 90 degrees, or the two ($\mathbf{B}$ field and the surface of coil) are perpendicular to each other and maximized when the cosine angle is 0 degrees.

The maximum magnetic flux that is passing through the tag coil is obtained when the two coils (reader coil and tag coil) are placed in parallel with respect to each other. This condition results in maximum induced voltage in the tag coil and also maximum read range. The inner product expression in Equation 5 also can be expressed in terms of a mutual coupling between the reader and tag coils. The mutual coupling between the two coils is maximized in the above condition.

**FIGURE 4: A BASIC CONFIGURATION OF READER AND TAG ANTENNAS IN RFID APPLICATIONS**
Using Equations 3 and 5, Equation 4 can be rewritten as:

**EQUATION 6:**

\[
V = - N_2 \frac{d\Psi}{dt} = - N_2 \frac{d}{dt} \left[ \vec{B} \cdot d\vec{S} \right] = - N_2 \frac{d}{dt} \left[ \frac{\mu_0 N_1 i_1 a^2}{2(a^2 + r^2)^{3/2}} d\Psi \right] = - \mu_0 N_1 N_2 a^2 \left( \frac{\pi b^2}{2(a^2 + r^2)^{3/2}} \right) \frac{di_1}{dt} = - M \frac{di_1}{dt}
\]

where:
- \( V \) = voltage in the tag coil
- \( i_1 \) = current in the reader coil
- \( a \) = radius of the reader coil
- \( b \) = radius of tag coil
- \( r \) = distance between the two coils
- \( M \) = mutual inductance between the tag and reader coils, and given by:

**EQUATION 7:**

\[
M = \frac{\mu_0 \pi N_1 N_2 (ab)^2}{2(a^2 + r^2)^{3/2}}
\]

The above equation is equivalent to a voltage transformation in typical transformer applications. The current flow in the primary coil produces a magnetic flux that causes a voltage induction at the secondary coil.

As shown in Equation 6, the tag coil voltage is largely dependent on the mutual inductance between the two coils. The mutual inductance is a function of coil geometry and the spacing between them. The induced voltage in the tag coil decreases with \( r^3 \). Therefore, the read range also decreases in the same way.

From Equations 4 and 5, a generalized expression for induced voltage \( V_n \) in a tuned loop coil is given by:

**EQUATION 8:**

\[
V_n = 2\pi f S Q B_0 \cos \alpha
\]

where:
- \( f \) = frequency of the arrival signal
- \( N \) = number of turns of coil in the loop
- \( S \) = area of the loop in square meters (m²)
- \( Q \) = quality factor of circuit
- \( B_0 \) = strength of the arrival signal
- \( \alpha \) = angle of arrival of the signal

In the above equation, the quality factor \( Q \) is a measure of the selectivity of the frequency of the interest. The \( Q \) will be defined in Equations 43 through 50.

**FIGURE 5: ORIENTATION DEPENDENCY OF THE TAG ANTENNA**

The induced voltage developed across the loop antenna coil is a function of the angle of the arrival signal. The induced voltage is maximized when the antenna coil is placed in parallel with the incoming signal where \( \alpha = 0 \).