

**Lost & Found Web Application**

Freddie Timmins

11341061

Industrial Development Project

Software Design & Development Higher Diploma (Industry Stream)

Table of Contents

1. Introduction
   1. Document Outline
   2. Document Description
2. Design Considerations
   1. Assumptions
   2. General Constraints
   3. Code Management
3. System Architecture
   1. Architecture Strategies
   2. SQL Server Database
   3. ASP.NET
   4. MVC
4. System Development
   1. Project Timeline and Key Challenges
   2. Development Methods
      1. Method Adopted
      2. Weaknesses
      3. Strengths
   3. Compromises
   4. Extensions
   5. Future Extensions
5. Detailed System Design
   1. Use Case Documentation
      1. Use Case Diagrams
      2. Use Case Specifications
   2. Domain Model
      1. Domain Model Class Diagram
      2. Domain Model Design Decisions
   3. Database Design
      1. Introduction
      2. Entity Relationship Diagram
      3. Tables & Attributes
      4. Design Decisions
   4. Code Breakdown
      1. Key Challenges
6. System Test Design
7. Bibliography
8. Acknowledgements
9. Appendices
10. **Introduction**
    1. **Document Outline**

The objective of this document is to inform the reader of what the application does and specify in detail how the application was developed. It documents the key challenges experienced by the student throughout the process as well as the testing carried out.

* 1. **Document Description**
     1. **Introduction**

For this project a lost and found web application was developed. The application was developed using Visual Studio 2013, where an ASP.NET application framework was used along with an MVC framework architecture. The primary programming language used was c#.

* + 1. **Objective of the Application**

The objective of this project was to create a lost and found web application. The application allows preregistered users to register their details, and registered users to log in in order to view the items page where they can post about items they have lost or found in an attempt to return lost items to their owners. Users can sort the posts on the items page by status and search from items by item type. It also allows users to view the Users page in order to easily filter through users who have allowed their details be displayed. The idea behind the user page is for an ease of access to contact information of other users.

* + 1. **System Overview**

Lost and Found Users

The lost and found web application was designed to be used by users of all skill levels. In particular the application was designed for people who had lost something they wanted to get back or people who had found something that they wanted to return.

Core Functionality

The main objective of the lost and found application was to provide a place for:

* Registering user details and determining the authorization levels of the user throughout the application.
* Users to post about lost and found items.
* Users to store their username and email address.

User Roles

In the lost and found application there are two defined users. The standard application user can visit the page and view the home page, about page and contacts page. In order to view the items page and user’s page they must first register their details. Once registered they can visit the items page and user’s page, create new items or users and view the details of each item or user. If the wants to edit the information in a post they must get communicate this to the admin.

The admin is the second role defined in the application. Like the standard user the admin must also log in but once logged in the admin has access to all methods. This means the admin can edit item and user posts and also delete item and user posts if necessary. The intention of the student is to use the delete method as little as possible in order to keep the posts as a history.

1. **Design Considerations**

In this section the main design considerations of the application will be discussed. Design considerations were an ongoing process throughout the duration of the project.

* 1. **Assumptions**

**End User**

It was assumed that the end user would have suitable amount of knowledge to be able to access the web pages and navigate through the application. In order for this to be achievable it was important to consider a simple and aesthetically pleasing layout.

It was also assumed that the user would have an email address in order to register their details to access the sites items page and users page.

**Admin**

The system admin was assumed to have a suitable amount of knowledge in order to navigate the application and be able to access user’s posts when required to update the details of the post.

* 1. **General Constraints**

A number of constraints were identified throughout the development of the application including:

* Standards Compliance

For this project the student developed an ASP.NET MVC web application in Visual Studio 2013 using C# as the coding language and a SQL Server database. This meant the student was obliged to comply with the constraints posed by these technologies.

* Security Constraints

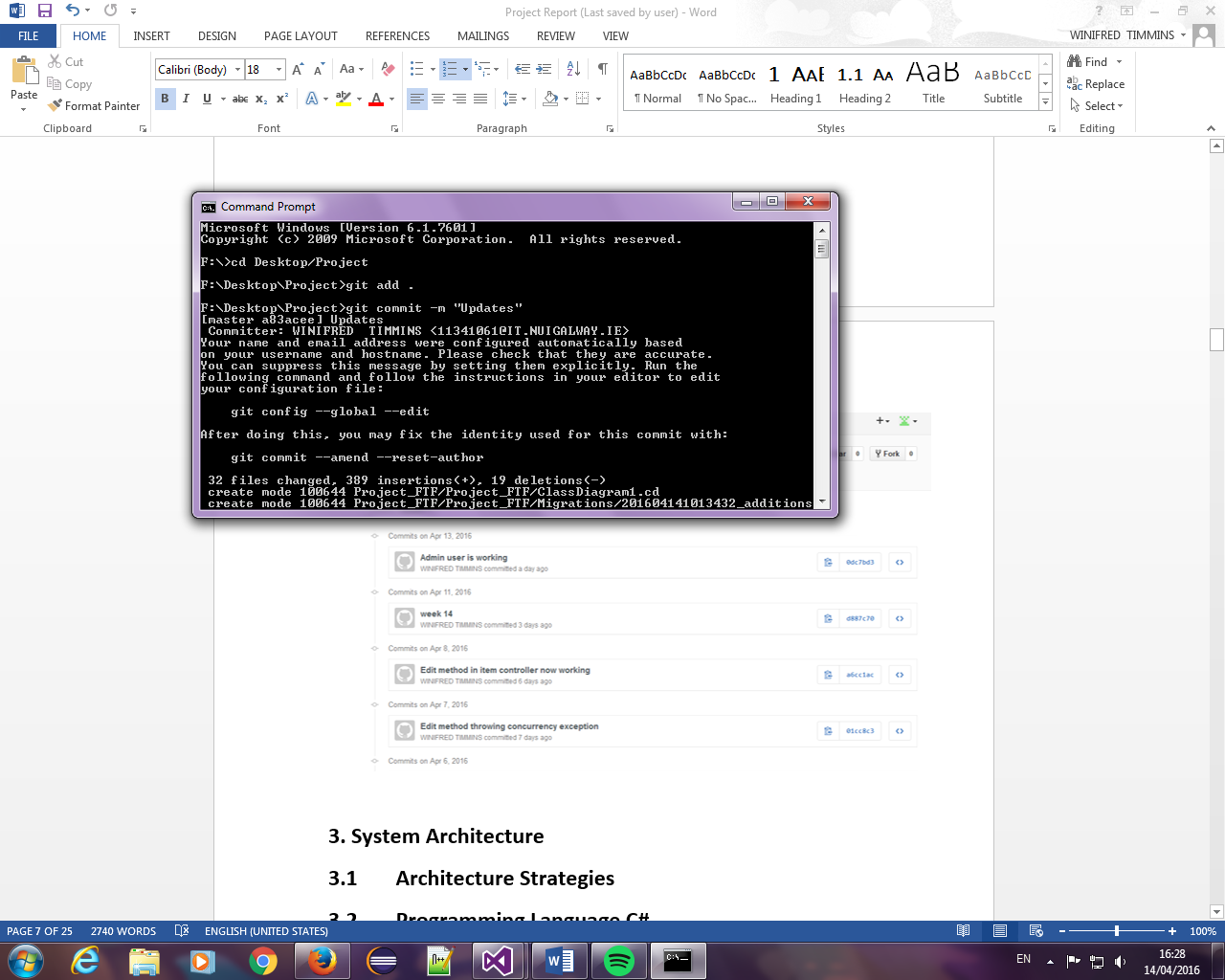
It was established that privacy needed to be maintained between users and posts. Each user must have a unique identity within the database.

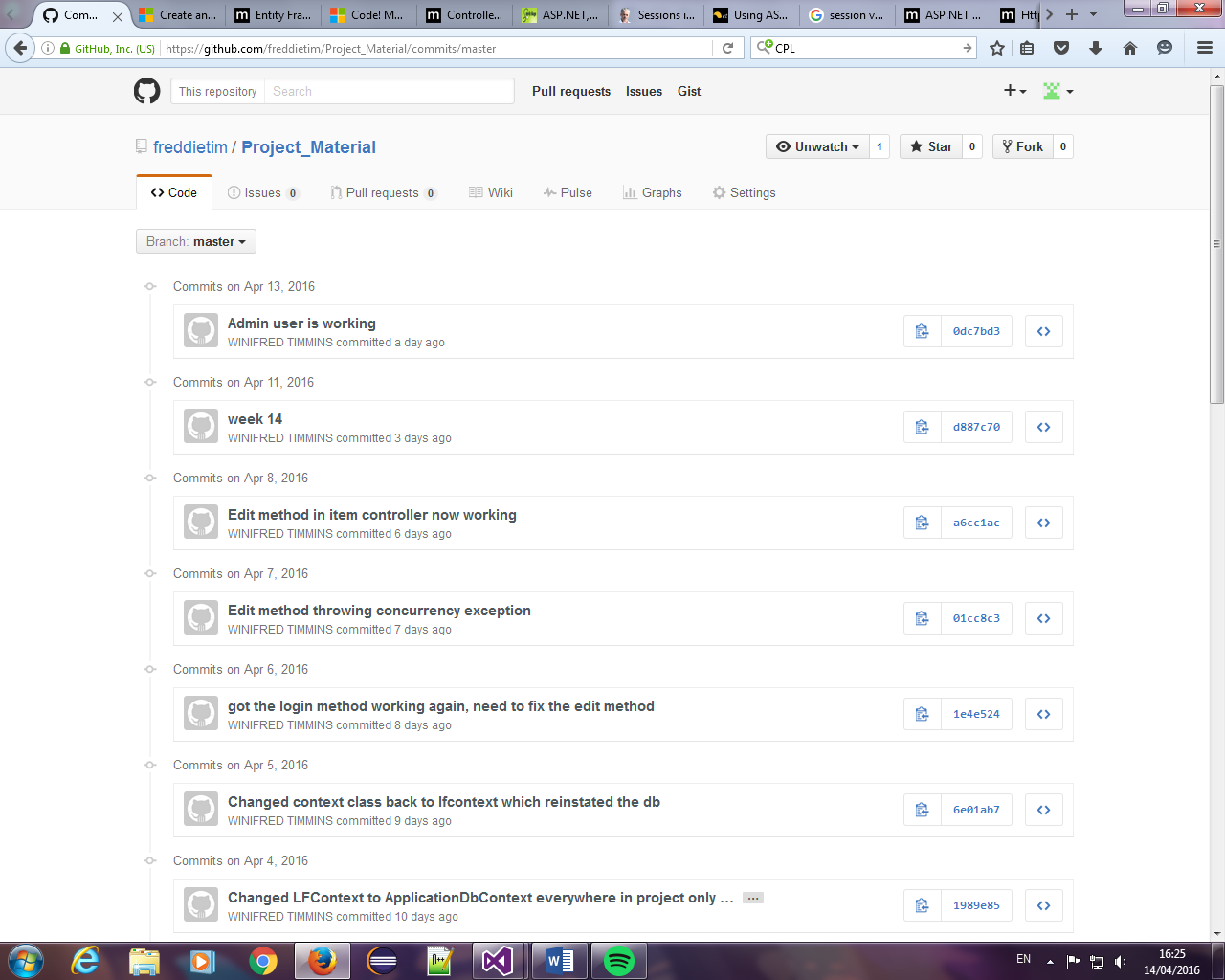
* Usability

It was extremely important that the application was simple to use and easy to navigate for users of all abilities.

* 1. **Code Management**

GitHub was used as the source control for this project. This allowed the student to commit developments throughout the course of the project and revert back to previous commits to examine and changes that had been made to the system. The student used the command line to push commits to GitHub, marking each one with a relevant message in order to conveniently search through them.





1. **System Architecture**
   1. **Architectural Strategies**

The architectural strategies decided upon for the development of this application are discussed in detail in this section. An ASP.NET MVC web application was developed with a SQL Server database using Visual Studio 2013 and the programming language C#.

* 1. **Programming Language C#**

This application was written using the programming language C#. The student was obliged to use C#, as the company they will be going to as part of their industry placement is a .NET company.

* 1. **SQL Server Database**

A SQL Server database was used by the student. It was developed using code first migrations with entity framework. Before enabling code first migrations every time a data model was changed it became out of sync with the database. This meant changes to any of the entity classes or context class resulted in the database being deleted and a new one created. For this application this was not very useful because it meant posts created by users were not being stored to the database. For this reason the student felt it was best to enable code first migrations. Anytime a change was made to the entity models or context class a migration was added in the package manager console and the database was updated. This meant that as users posted about new items/users the information was being stored in the database and no information was being lost.

The context class was another important feature of entity framework as it linked the models to the database using a connection string.

* 1. **ASP.NET**

ASP.NET was the web application framework used by the student to create this project. ASP.NET is a platform for developing and running applications on a web server. It allowed the developer to create an application with access to a variety of classes in the .NET framework which have the added benefits of type safety, inheritance etc. It uses html as the mark-up language.

* 1. **MVC**

MVC was the framework architecture adopted to create this web application. MVC is comprised of three different components, the model, the view and the controller.

The model corresponds to the business layer and represents the state of a particular entity in the application. The view corresponds to the display layer. It receives any necessary information from the controller and provides a user interface that displays that information. The controller corresponds to the input controls. It deals with any interactions and updates the model to echo any changes in the state of the application, and then feeds this information to the view. This architecture was very useful as it allowed the student to access individual parts of the application without interfering with other classes, controllers or views. Once a model is created a controller is then scaffolded with basic CRUD functionality and the corresponding views are also created in the application solution.

1. **System Development**

This section displays the project timeline including a list of the key challenges dealt with each week during the project.

* 1. **Project Timeline & Key Challenges**

Week 1

In week one the student got in touch with the company to find out if they had any project specifications they would like the student to complete.

**Key Challenge:**

The key challenge in week one was deciding on what the project would be.

Week 2

The company informed the student that they would not give the student a specific project topic and to proceed with the student’s own idea.

Week 3

Key challenge from week one was overcome and a project was agreed upon between with the supervisor. It was decided to create a web application for a lost and found service.

**Key Challenge:**

The key challenge faced this week was to figure out how exactly to go about creating a web application using c# and which sources would be most relevant to help in the development of the application.

Week 4

In week 4 it was decided to develop the application using an ASP.NET framework along with an MVC framework architecture.

**Key Challenge:**

The key challenge in week 4 was determining what exactly the application needed in terms of pages.

Week 5

In week 5 a user model, user controller and corresponding views were created.

**Key Challenge:**

The key challenges this week were learning about how the models interacted with the controllers and views.

Week 6

In week 6 sorting and filtering were added to the user page and work was carried out the format of the application.

**Key Challenge:**

The key challenges in week 6 included figuring out how users would be able to post about items they have found or lost and how that information would be displayed in the application.

Week 7

In week 7 the student learned about connection strings and how these would be useful in connecting up the database. The database was created using code first migrations.

**Key Challenge:**

The key challenges in week 7 included determining which tables to use in the database and how users would post about items, would they post only found items or both lost items and found items.

Week 8

In week 8 more work was carried out on what models were most appropriate to use in the application.

**Key Challenge:**

The key challenges of week 8 were figuring out how the user would interact with the system and what methods they would have access to.

Week 9

In week 9 different functionality was added to some of the view pages to see what worked best with the application.

**Key Challenge:**

The key challenge in week 9 was determining what added functionality was necessary and worked best with the application.

Week 10

In week 10 work was carried out on editing the layout of the application and on the site css.

**Key Challenge:**

The key challenge in week 10 was creating new items and users and successfully adding them to the database.

Week 11

In week 11 the student finally decided on what tables would be best for the database. One table for items was decided on with the relevant rows for a user to obtain enough information about the item.

**Key Challenge:**

The key challenge in week 11 was adding authorization and authentication to the application.

Week 12

In week 12 it was decided that user’s should only have access to the create method and details method on both the items page and the users page.

The admin could then have access to all methods and users must get in touch with the admin to update the status of a post.

**Key Challenge:**

The key challenge this week was getting the admin user to work.

Week 13

In week 13 the built in ApplicationDbContext class was removed and only one context class was used the LFContext class.

**Key Challenge:**

The key challenges in week 13 included dealing with database concurrency exceptions and adding session objects to the application. Instead of the admin accessing the edit method a user should be able to edit their own posts only.

Week 14

**Key Challenge:**

* 1. **Development Methods**
     1. **Method Adopted**

The student opted to develop the application using ASP.NET MVC which allows for a test driven development approach. It allowed the student to develop different aspects of the system without having to refactor or depend on another element.

* + 1. **Weaknesses**

During the intermediate phases of development using a framework meant the student learned the framework as opposed to the intricate detail of the language. As the semester progressed the student learned more about the language and specific design features of the framework such as razors, controllers and how the individual components of the application interacted.

* + 1. **Strengths**
  1. **Compromises**
  2. **Extensions**

Certain extensions were made to the application that were not specified in the requirements document. This included a google map where users could enter the location the item was found or lost in and it would be highlighted on the map.\*\* The student also added a graph to the application to compare the location where the item was found with the item type. \*\*

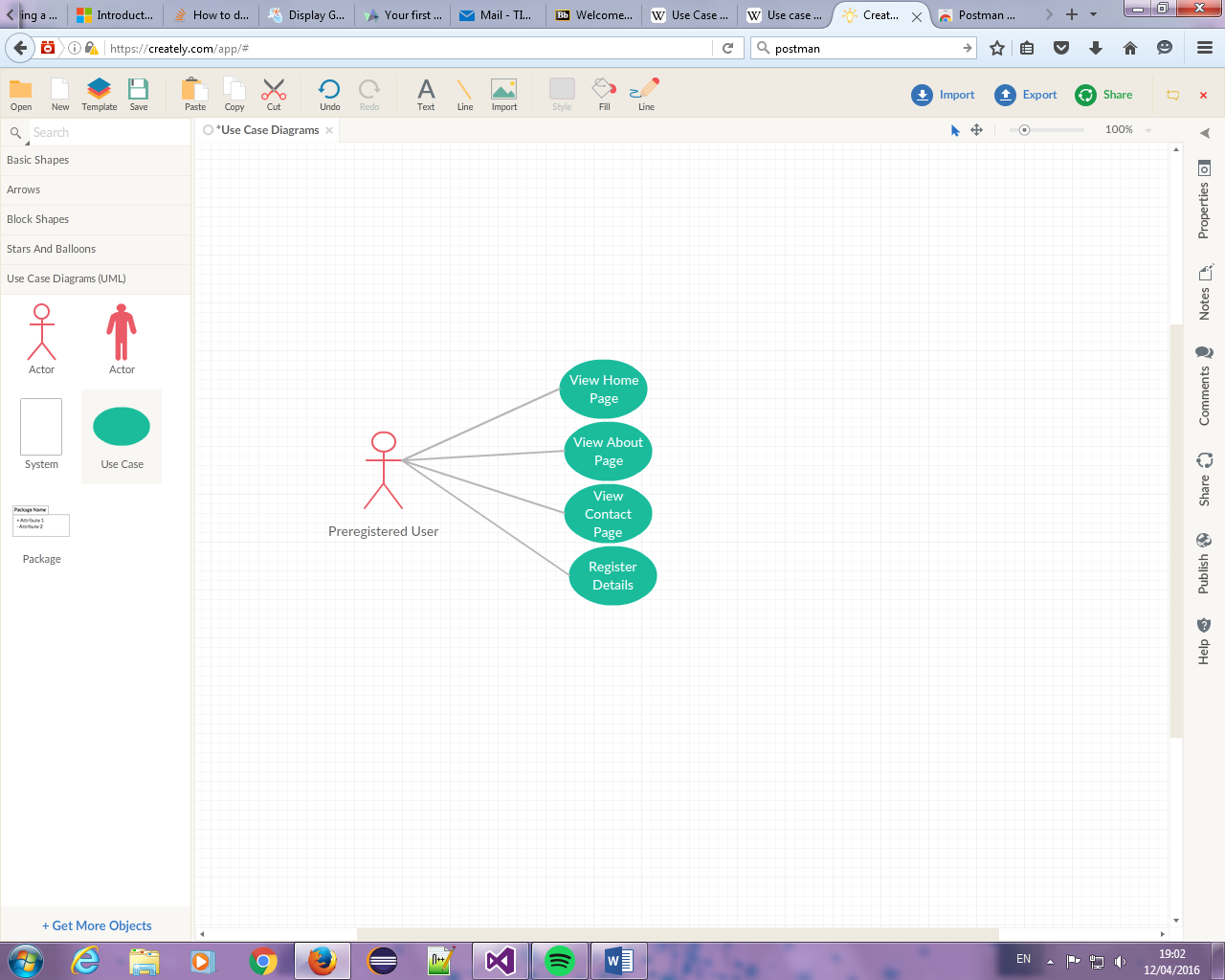
* 1. **Future Extensions**

Given the time frame of the project some desirable functionality was not included in the application. Had the student more time session variables would have been implemented to allow users log in and edit their own posts without the need for the admin. The student would also have hosted the application on the cloud to further the development of the application.

1. **Detailed System Design**
   1. **Use Case Documentation**
      1. **Use Case Diagrams**

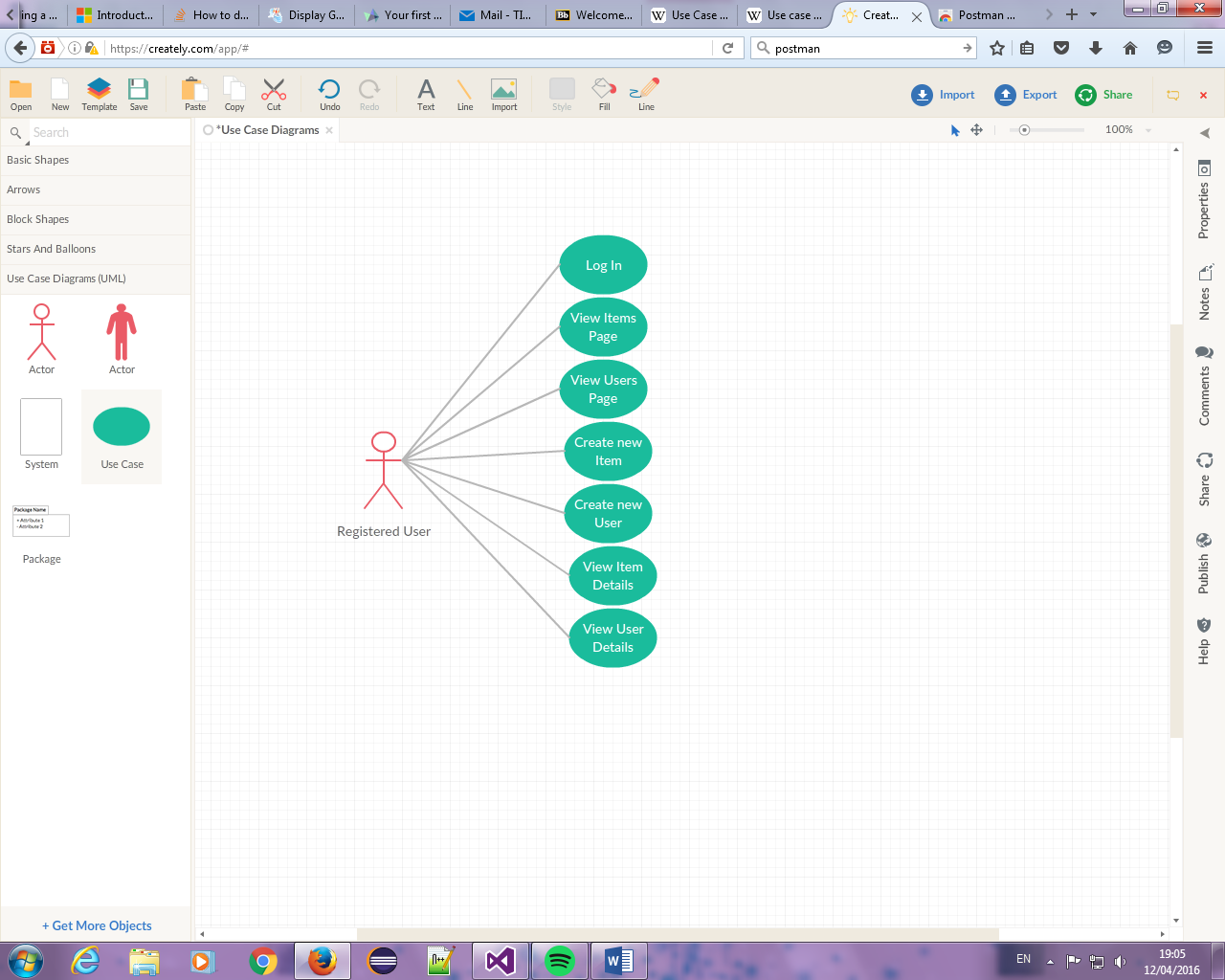
Preregistered User

The preregistered user can only interact with a certain number of the application pages before they must register their details. They can visit the application homepage, the about page, the contact page and the register details page.



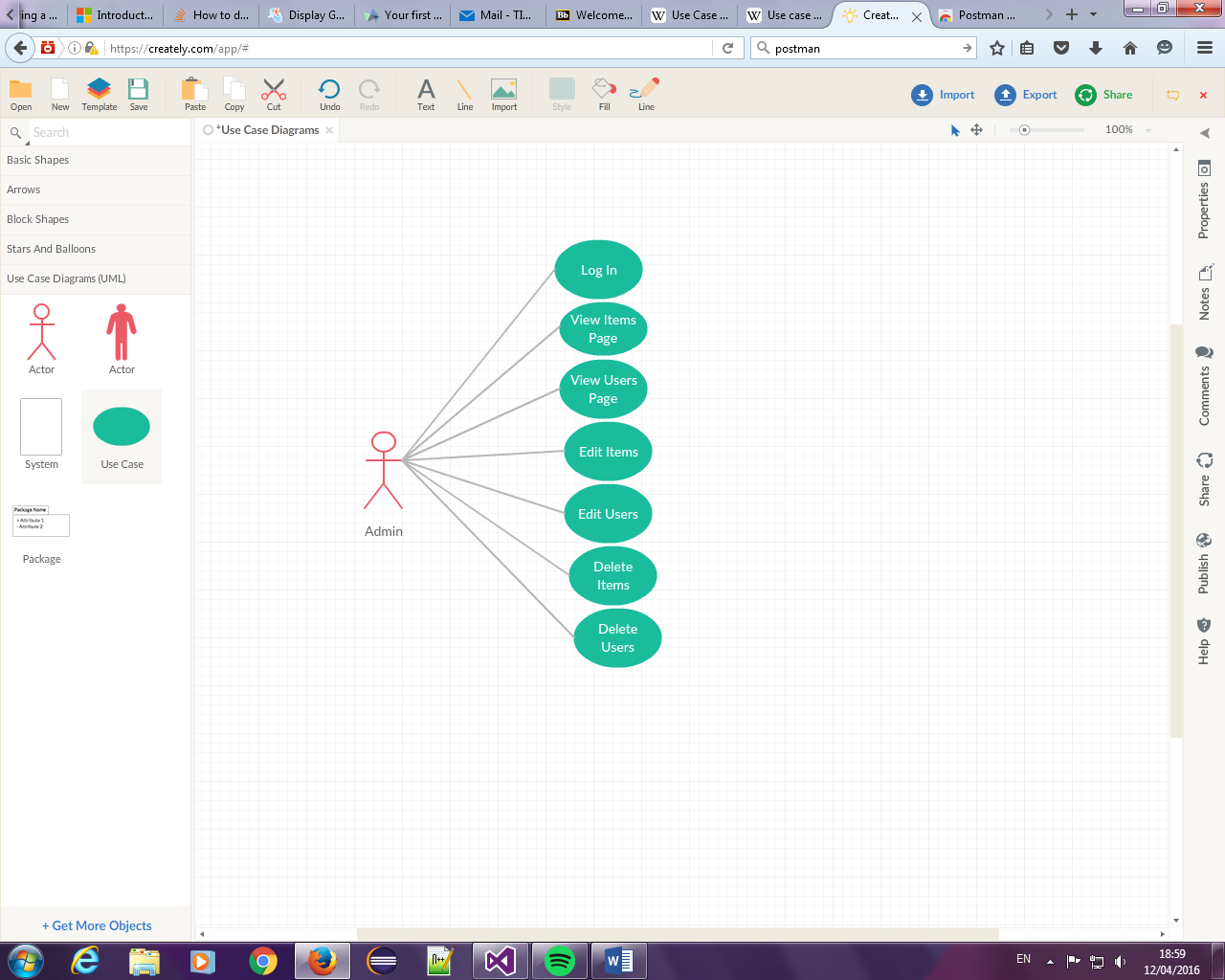
Registered User

Once the user is registered they can access all of the application pages. They can now view the items page where all of the posts about lost and found items are, create new items and view the details of posts on the page. They can also visit the users page to view the details of other users of the application, and create new users. They can sort through both items and users and search through items and users by specifying item type and user name.



Admin

The admin interacts with the system with the same functionality as the registered user but they also have access to the edit method and delete method. When users want to change the information on the post of the items page they must contact the admin who can complete the update for them.



* + 1. **Use Case Specification**

**Use Case Name:** Register Details

**Purpose:** Allows preregistered users to register their details in order to have access to the application.

**Actors:** Preregistered Users

**Course of Events:**

1. Preregistered users visit the application homepage. Here they can access the about page and contact page but must register in order to access the items page and users page.
2. Preregistered users enter an email address, password and password confirmation.
3. Registration successful, users can now access the items page and users page.

**Conditions Triggering Alternative Flow:**

**Alternative Course of Events:**

**Preconditions:**

Use Case Name: Log In

Purpose: Allows registered users to access all application pages.

Actors: Registered Users, Admin

Course of Events:

1. Users navigate to log in page and enter email address and password.
2. Successful login allows user access all application pages.

**Conditions Triggering Alternative Flow:**

**Alternative Course of Events:**

**Preconditions:**

Use Case Name: Create Item

Purpose: Allows users to post about items they have lost or found.

Actors: Registered Users

Course of Events:

1. User navigates to the Items page.
2. User clicks the Create New button.
3. Form appears in front of the user

**Use Case Name:** Register Details

**Purpose:** Allows preregistered users to register their details in order to have access to the application.

**Actors:** Preregistered Users

**Course of Events:**

1. Preregistered users visit the application homepage. Here they can access the about page and contact page but must register in order to access the items page and users page.
2. Preregistered users enter an email address, password and password confirmation.
3. Registration successful, users can now access the items page and users page.

**Conditions Triggering Alternative Flow:**

**Alternative Course of Events:**

**Preconditions:**

**Use Case Name:** Register Details

**Purpose:** Allows preregistered users to register their details in order to have access to the application.

**Actors:** Preregistered Users

**Course of Events:**

1. Preregistered users visit the application homepage. Here they can access the about page and contact page but must register in order to access the items page and users page.
2. Preregistered users enter an email address, password and password confirmation.
3. Registration successful, users can now access the items page and users page.

**Conditions Triggering Alternative Flow:**

**Alternative Course of Events:**

**Preconditions:**

**Use Case Name:** Register Details

**Purpose:** Allows preregistered users to register their details in order to have access to the application.

**Actors:** Preregistered Users

**Course of Events:**

1. Preregistered users visit the application homepage. Here they can access the about page and contact page but must register in order to access the items page and users page.
2. Preregistered users enter an email address, password and password confirmation.
3. Registration successful, users can now access the items page and users page.

**Conditions Triggering Alternative Flow:**

**Alternative Course of Events:**

**Preconditions:**

* 1. **Domain Model**
     1. **Domain Model Class Diagram**
     2. **Domain Model Design Decisions**
  2. **Database Design**
     1. **Introduction**

The database was designed using code first migrations with entity framework.

* + 1. **Entity Relationship Diagram**
    2. **Tables and Attributes**
    3. **Design Decisions**
  1. **Code Breakdown**
     1. **Key Challenges**

1. **System Test Design**
   1. **Testing Overview**

Software testing is an extremely important part of the development process. It allows the developer to evaluate code early and change anything that needs to be changed without potentially harming the rest of the system.

The student tested the system to ensure

* It met the requirements of the design specification
* It responded appropriately to various kinds of input
* It was easy to use and achieved the required result of the end user.
* It performed tasks within an acceptable time frame.

This was done using both black box testing and white box testing.

* 1. **Test Framework**

In order for the student to carry out testing a suitable testing framework had to be established. The testing framework that supports unit testing for the C# programming language and all .NET languages is NUnit. NUnit provides a console runner which displays a solid red line for tests that have failed and a solid green line for tests that have passed. For the student to complete the unit testing creating a new class library in the solution explorer was required. Within this class library two new test classes were also created.

* 1. **White Box Testing**

White box testing, unlike black box testing, looks at the internal structure of the application. White box testing was used in the application to test the main functionality of the CRUD methods. The CRUD methods include a create method, read method, update method and delete method and were used in both the items page and the users page.

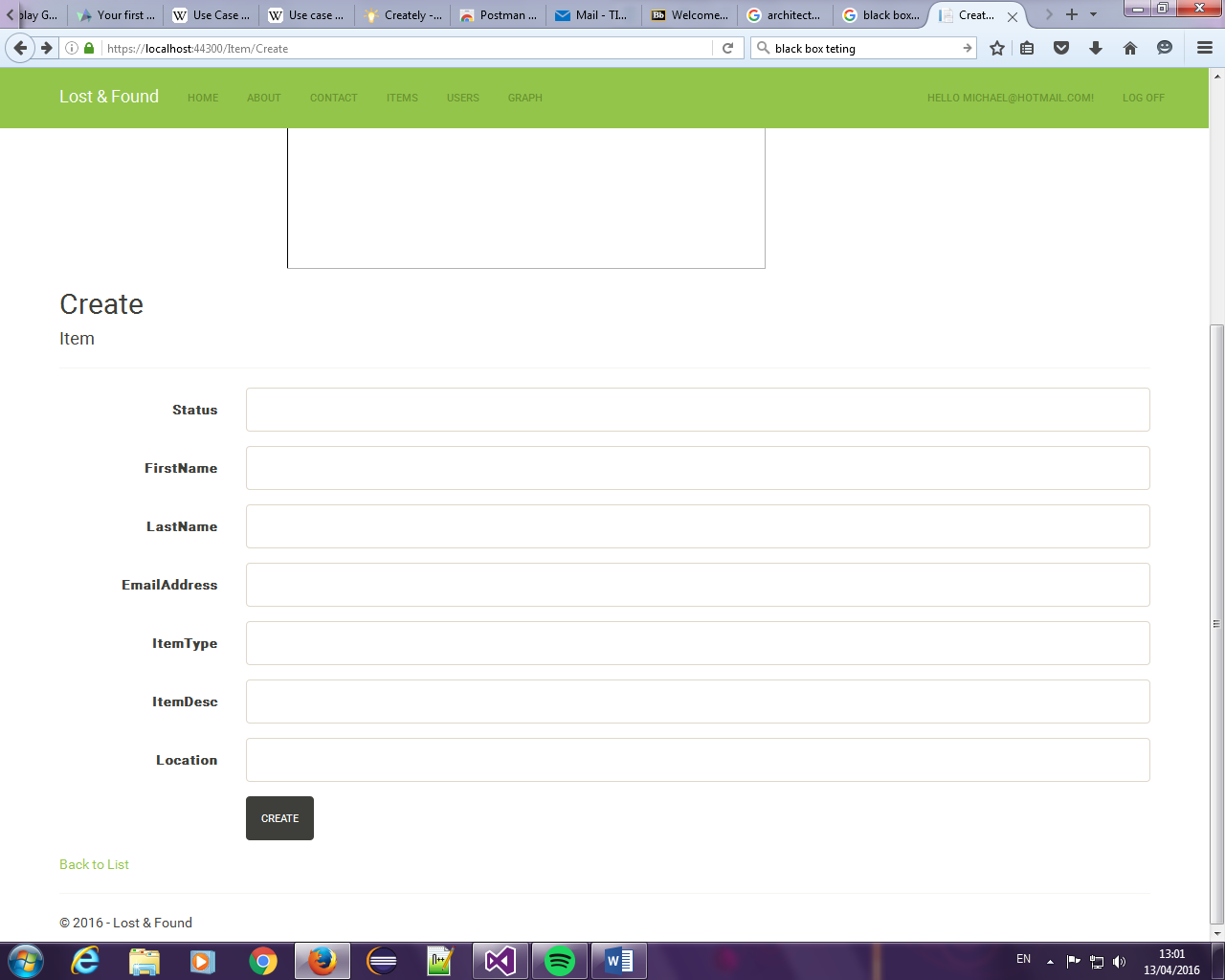
* 1. **Black Box Testing**

Black box testing is used to examine the functionality of an application without delving into the internal workings of the application.

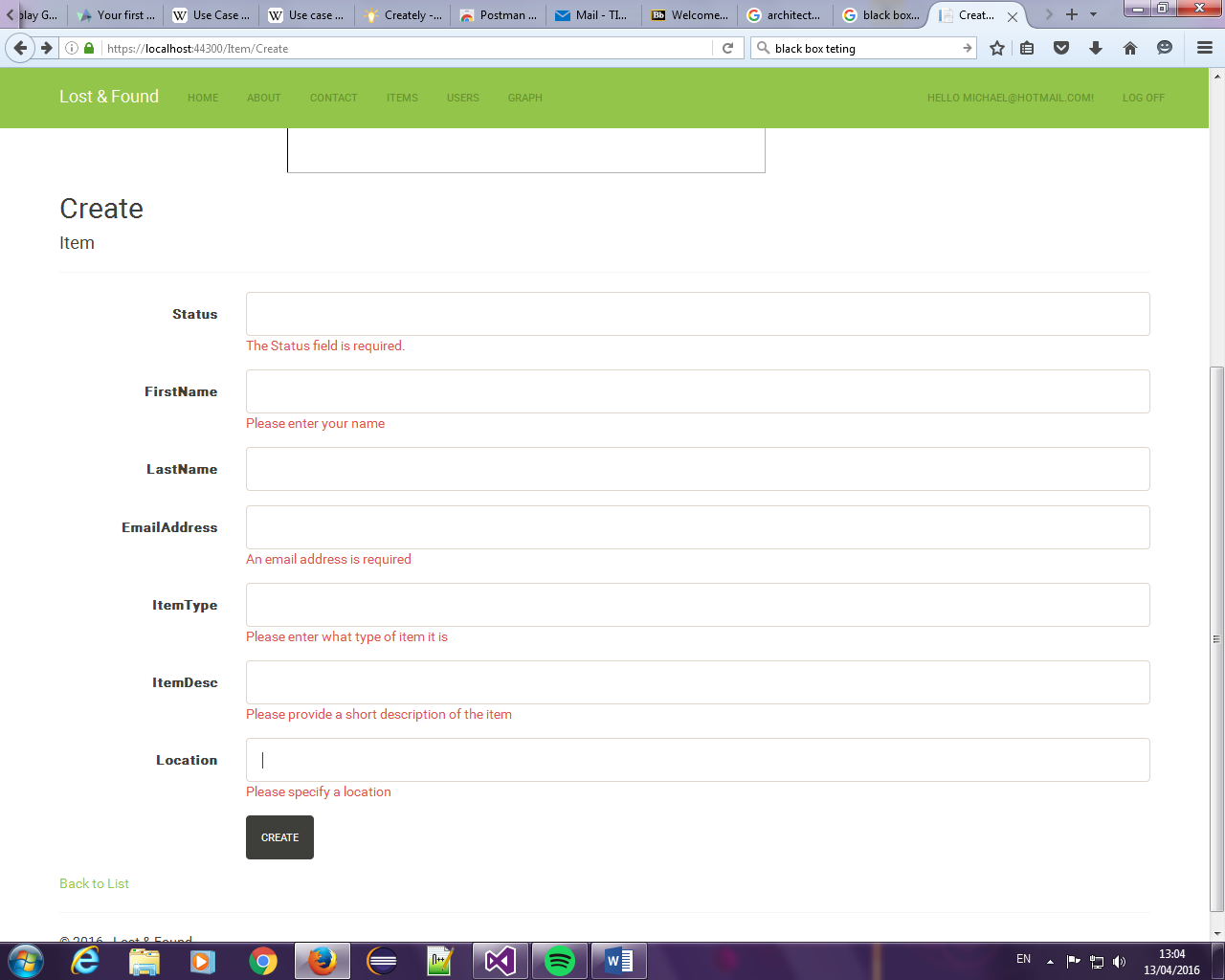
In this section examples of black box testing are shown through form validation.

* Create new item

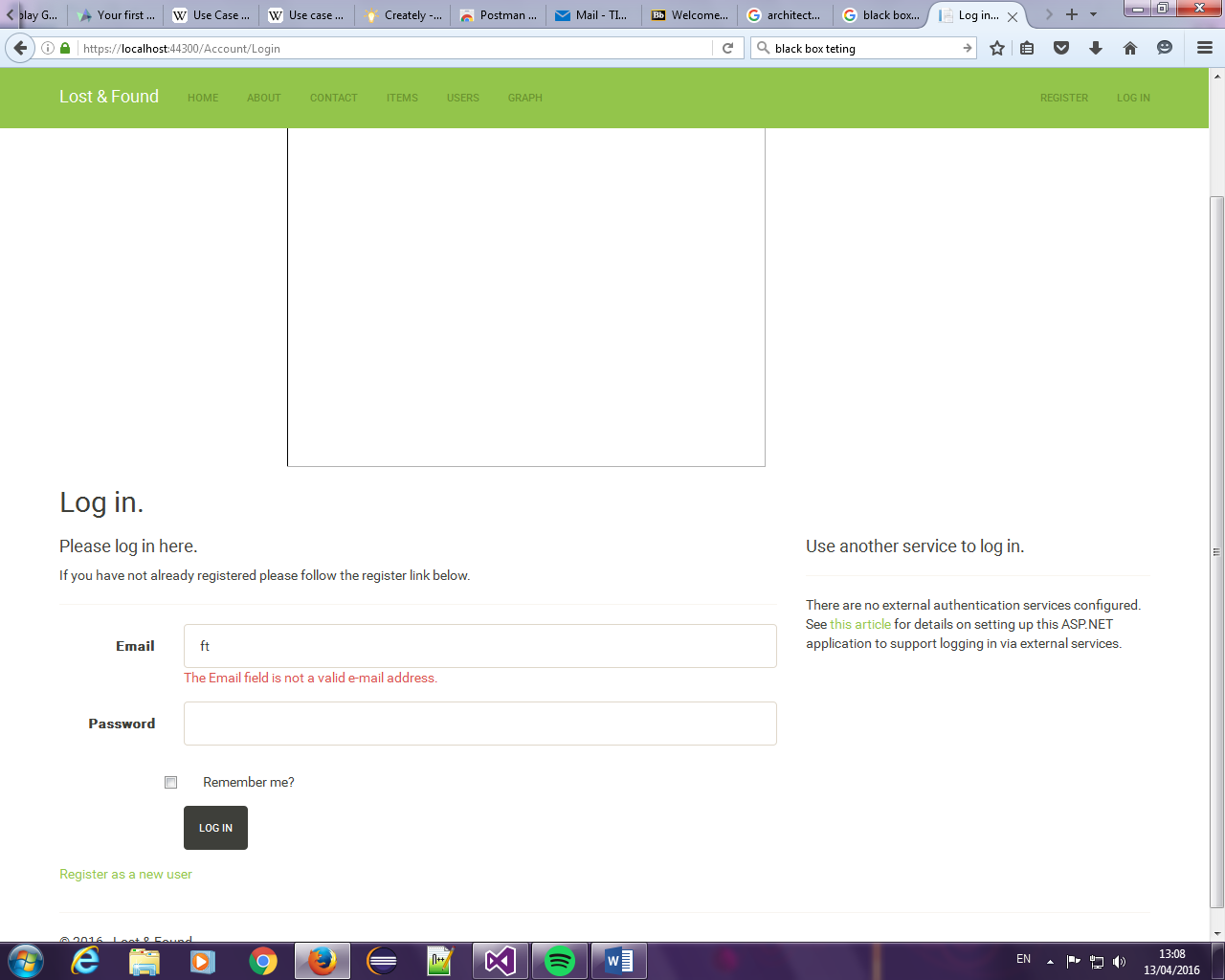
Looking at the Items page, in order to create a new item the user must fill out the ‘create new’ form. As long as a user is registered they can post about items they have lost or found.



If the user had filled out the form incorrectly, when they clicked the create button, prompts appear under each of the boxes and advised the user on what was required.

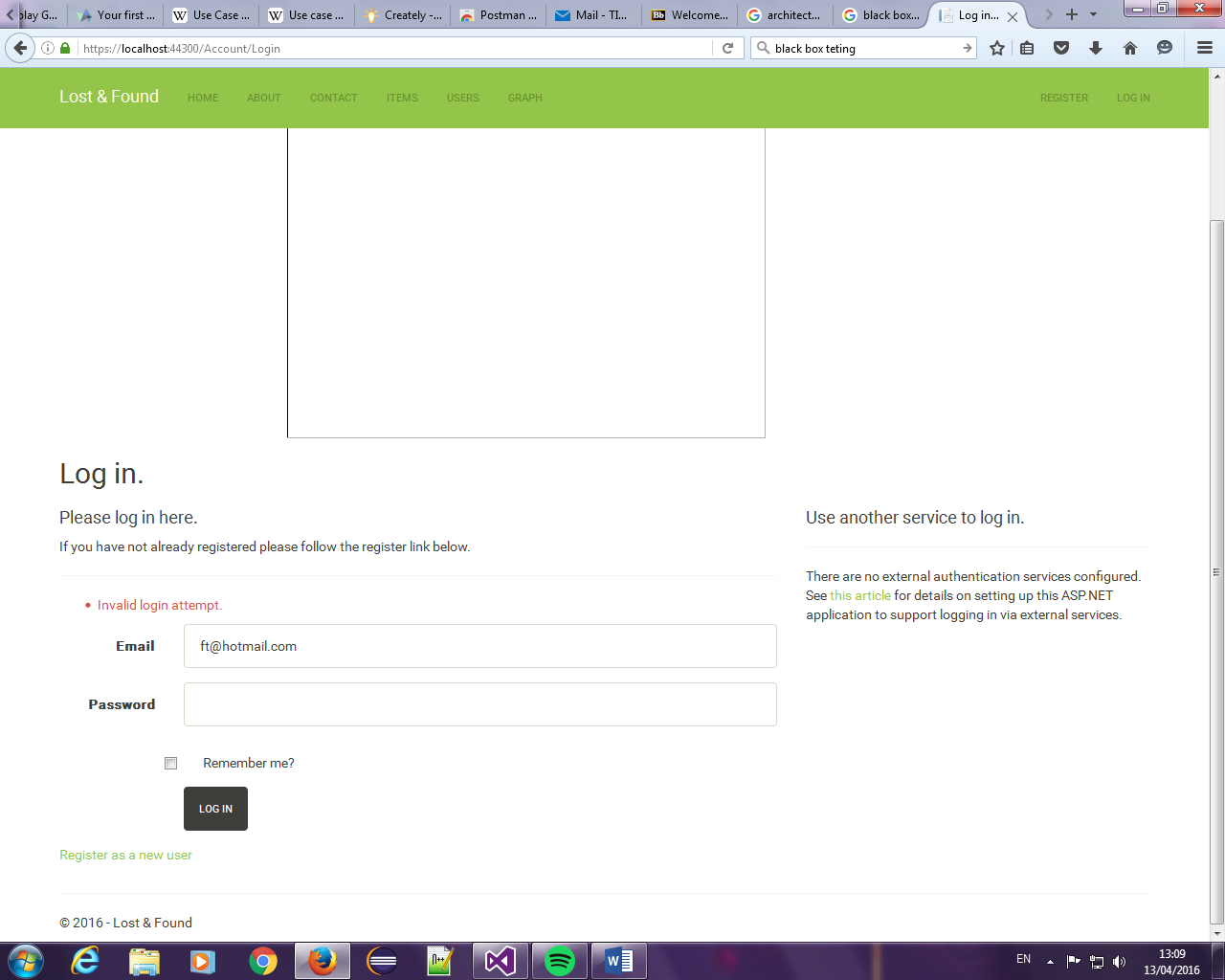


* Failed log in attempt

Here the user has navigated to the log in page. An invalid email address was entered. When the user clicked the log in button the prompt appeared to inform them to enter a valid email address.

* Preregistered User

Here a user who was not registered attempted to log in. The log in page informed the user that the log in attempt was invalid. The user must register their details before logging in.



* Password and Confirmation password do not match

The user went to the register page and attempted to register their details. The password and confirmation password were not the same. When the user clicked ‘register’, the prompt appeared on the screen to inform the user of the problem.

