National College of Ireland

BSc in Computing

2016/2017

Name Surname

Kevin Moss

Student ID

X13562797

Full Title

Technical Report

Advanced Image Authentication Level

National College of Ireland

Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: Kevin Moss
Student ID: **X13562797**
Supervisor: **Sara Kadry**

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarized overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: Kevin Moss

Date:08/05/2017

Table of Contents

Executive Summary

The problem that my system has been designed to address is that of data security i.e. the protection of sensitive data from unauthorised access, modification or deletion both in a business environment and for personal users. Different forms of data are precious to different individuals for different reason such as family photos, business secrets, medical records, financial data and user credentials for sensitive system e.g. government or state run institutions.

The technical solution implemented by my project to achieve these goals is to create a secure MVC application within visual studio 2015 using my knowledge of secure coding applications and penetration testing to create a locked down base application within which to create my advanced authentication level. The system I have developed works by having a user select and image they wish to assign to their user id, only they know which image this is. Once the user selects their image the metadata of that image is read and uploaded to the database. This metadata is then encrypted and held within the database table along with the user-id. Hashing is also implemented along with the encryption to ensure that upon logging in the user will need to select the their “Trusted Imaged” or the hashed metadata will mismatch and not allow the user to login.

The final evaluation of my system showed that it has a good peak and average response time ensuring that it has a high level of availability, the cryptography and hashing is implemented to a high standard ensuring the confidentiality of the data held within the database. The entire web application has been developed using secure coding principles as recommended by OWASP ensuring a reduced and hardened attack surface to maintain the application’s Integrity.

1 Introduction

1.1 Background

Cyber security can be defined as the protection of systems, networks and data in cyberspace, this is a critical issue for both the world of business and the average user of any smartphone, fitness band, or any device that is connected to the ever-expanding world of technology we find ourselves in today, a new and important emerging market in this world of technology is the “Internet of things” or (IOT), this market ventures to bring the internet into our homes, schools and critical infrastructure used in the running of our everyday lives. If this is to be the case, then we will want to know that any of these systems that we have put our faith and trust in will be secure from malicious attacks. The intended customers are health care facilities, government institutions, and public service institutions, or any individual requiring another level of security beyond multi-factor authentication for protecting sensitive information.

Cybersecurity Ventures predicts global annual cybercrime costs will grow from $3 trillion in 2015 to $6 trillion annually by 2021, which includes damage and destruction of data this can be from priceless photos of memories with loved one to college work that has taken many hours to business secrets that will never be returned, stolen money, lost productivity, theft of intellectual property, theft of personal and financial data, embezzlement, fraud, post-attack disruption to the normal course of business, forensic investigation, restoration and deletion of hacked data and systems, and reputational harm.

One of the best-known breaches of personal information and customer data was the Ashley Madison data breach that hit the infamous infidelity dating site back in the summer of 2015. A group of attackers discovered weaknesses in password encryption and used these to crack the crypt-hashed passwords. The criminals responsible gained the personal details and credit card numbers of over 11 million users, this data was then sold for a price on the darknet. The company has since lost its CEO and now faces lawsuits and other legal battles due to a lack of sufficient security.

The mobile phone provider TalkTalk was the target of a group of teenage hackers who stole the details of over 20,000 customers. The hackers were quickly identified and dealt with, but the company has been left with a bill of up to £35 million, having had millions wiped off its share price, and is facing law suits from customers and investors. This goes to show that it may not necessarily need to be the most skillful or well orientated attack the consequences for the victim can still be immense

Considering the impact a breach of security can have, I have decided to develop a new level of security in the form of an advanced authentication system using images, this project has presented with me with some challenges as I am new to the world of cyber security, it will be an opportunity to see what can be done to mitigate against attacks and to work with some new technology's and development environments such as visual studio and azure which will allow me to publish my application to the cloud. I am also looking forward to mastering a new programming language C# for this project, I believe it will be both challenging and rewarding experience.

The data held by these institutions are of the utmost importance as they effect the financial stability of businesses, anonymity of the internet, wellbeing and health of individuals, how the country is run and provide support for those within their jurisdiction, the confidentiality, integrity and availability of these institutions systems will need to be should be maintained to the highest standards at all times and my system will go towards ensuring this.

1.2 Aims

The purpose of my project is to develop an advanced authentication system using a user image as verification method that validates user identity for accessing applications such as Dropbox, Mobile Banking Apps, or any application containing sensitive data for the user / company.

The project involves two level of verification which will provide an increased level of security. Most of passwords available today can be broken. Hence this project is aimed to achieve the highest security in authenticating users to their desired application or web service containing sensitive information. The two levels are:

- First Level: The first level is a password system i.e. text based password, passphrase & multi-factor authentication to authenticate the user's credentials.
- Second Level: The second level is an image based where users can upload their image and then create a hash code of this image and assigning it to the user's account. During login process the system will automatically verify the new image hash code with the previous that is stored as "Trusted Image".

If the hash code of the image being used does not match the hash code of the stored "Trusted image" then the user will be unable to gain access to the system. The user will have two attempts if on the second attempt they are again unsuccessful the user must log a ticket to have all levels of their security reset including the "Trusted image".

I will be carrying out this project as a proof of concept to demonstrate its feasibility and the demonstrate that in principle this image authentication level method can be applied to any application or web-service to name but a few systems that will require the highest of security standards to maintain their confidentiality, integrity and availability at all times as these are the three-key principle of cyber security systems.

Overall the advanced authentication level will be easy to install and implement on any existing system or work as a key building block for a developing system. The system will be user friendly and require little training as the primary functions of the authentication is done behind the scenes and requires no input from the ender user aside from selecting their image and remembering their chosen "Trusted Image". Ideally the system will be ready for industry across multiple devices at the end of development and can begin deployment immediately.

The web application I am developing to host the advanced authentication level will address the OWASP top ten web application vulnerabilities and have countermeasures to address the highest ranked vulnerabilities to ensure the application is secure and a viable test environment for my system.

To address the OWASP top ten I will be addressing security misconfiguration through the implementation of custom error pages so not to reveal source code or any background information to the end user if a failure of error occurs.

Cross site request forgery is also highly ranked by OWASP to prevent this, I have implemented AntiForgeryTokens When we add AntiForgeryToken helper on View it creates a hidden field and assigns a unique token value to it and meanwhile a session Cookie is added to the browser. When we post form HTML it checks for RequestVerificationToken Hidden field and whether RequestVerificationToken Cookie are present or not. If either the cookie or the form RequestVerificationToken Hidden field values are missing, or the values don't match, ASP.NET MVC does not process the action.

Cross site scripting (XSS) is the input of malicious content into input fields and is widely used to retrieve sensitive data or have user credentials revealed allowing the malicious user to gain authentic credentials and masqueraded as a valid user on the system and cause severe damage. To avoid this attack we can set string query lengths on all input fields and escape special characters to ensure malicious content will either be denied or converted into usable content.

SQL Injection is the insertion of malicious SQL queries to reveal sensitive data held in the database of an application to a malicious user. The prevention of this attack can be done by using stored procedures to carry out database requests and functions where is possible, also the use of a whitelist for allowed symbols and string length validation will prevent confidential data being exposed to the malicious user. I will investigate the OWASP top 10 in greater detail later in the report.

1.3 Technical Approach

I will be approaching this project as a relative new comer to the cyber security field, I will firstly have to research the Software Development Life Cycle in Terms of Security, I will also be using the Secure development lifecycle to make sure my project is a highly secure as possible before I begin to code it.

I will draft up use cases so that I can check every possible scenario in which a user can interact with my project, this will include a login/logout scenario, a forgotten password / lost image scenario and various other use cases depending on how the project is being implemented.

As this is a proof of concept project, the documentation will be very thorough and all avenues of how this application can be applied or attacked will be investigated and documented. I will then need to research the various topics of the project I am not yet familiar yet, I will be researching various image hashing algorithms such as AES256 and the newest form of encryption homomorphic encryption.

Other concepts are new to me such as storing an allowing the image to be kept in a secure SQL database as a "Trusted image".

I will also be looking into different security options with my entire project, I will be applying secure coding to the entire application so as to prevent cross site scripting and broken authentication and session management, I will be locking down my database to prevent SQL injections. Encryption will also be heavily featured in all areas of my project.

Once all areas of my application and security features are completed, I will be applying threat modelling and penetration testing, in an attempt to evaluate the security of my application by safely exposing any weaknesses, I will then log these weaknesses and return to correct them and test the project once again until all weaknesses have been dealt with resulting in a secure application.

1.4 Technologies

I will be using Visual Studios 2015 as my coding environment. Visual studio is linked with Microsoft azure, I have created a Microsoft azure account which will allow me to create an SQL database and server in the cloud to hold my "Confidential data" the link with azure will also allow me project to be published in the cloud upon completion.

The project will be built using the C# coding language and there will be secure coding throughout the project. I will be keeping versions and instances of my project as I go along using "GitHub" and "GitBash" to allow any rollbacks necessary and to always have a backup of my project in case of failure or corruption of the project.

I will be using the "MVC" template in visual studio as it will allow me to have a good starting point in terms of security. Multi factor authentication in the project will be implemented using "Send grid" an API that will allow me to verify my user's using both text and e-mail. The advanced authentication level that I am designing myself will be implemented by taking an image and reading the meta data of this image using an image reader algorithm using stored procedures, the metadata will be converted into hash codes and stored in the SQL database, plain text used to describe the metadata will be encrypted using 256 AES encryption standards. The OWAP top 10 for web applications will also be addressed throughout the project using inline code secure coding techniques.

1.5 Structure

The report is structured in such a way that by reading the first section of the document the reader will be able to get a general overview as to why the project has been developed and exactly what the project aims to do and the problems it aims to address. I have provided examples of where security has failed and highlighted the problem that this can cause for the victim, I have described how my project will aim to provide an extra level of security for web applications and for mobile applications. I have also described the technologies that will be used to develop my project to completion.

In the second section of this report, I will highlight the requirements of my system, these will range from the functional requirements that will describe specific behaviour and functions of my system. I will also describe the data requirements to show the reader what data I will be working with SQL etc. The user requirements will be highlighted describing what the user expects to be able to do with the system. Environmental requirements, what operating systems my system can run on and if there are any pre-requirements to running the software. Usability requirements will also be captured.

I will also present use Cases for my major system requirements showing their scope, description, pre-condition, activation, main flow, alternate flows, exceptional flows, termination, post conditions and their frequency of use.

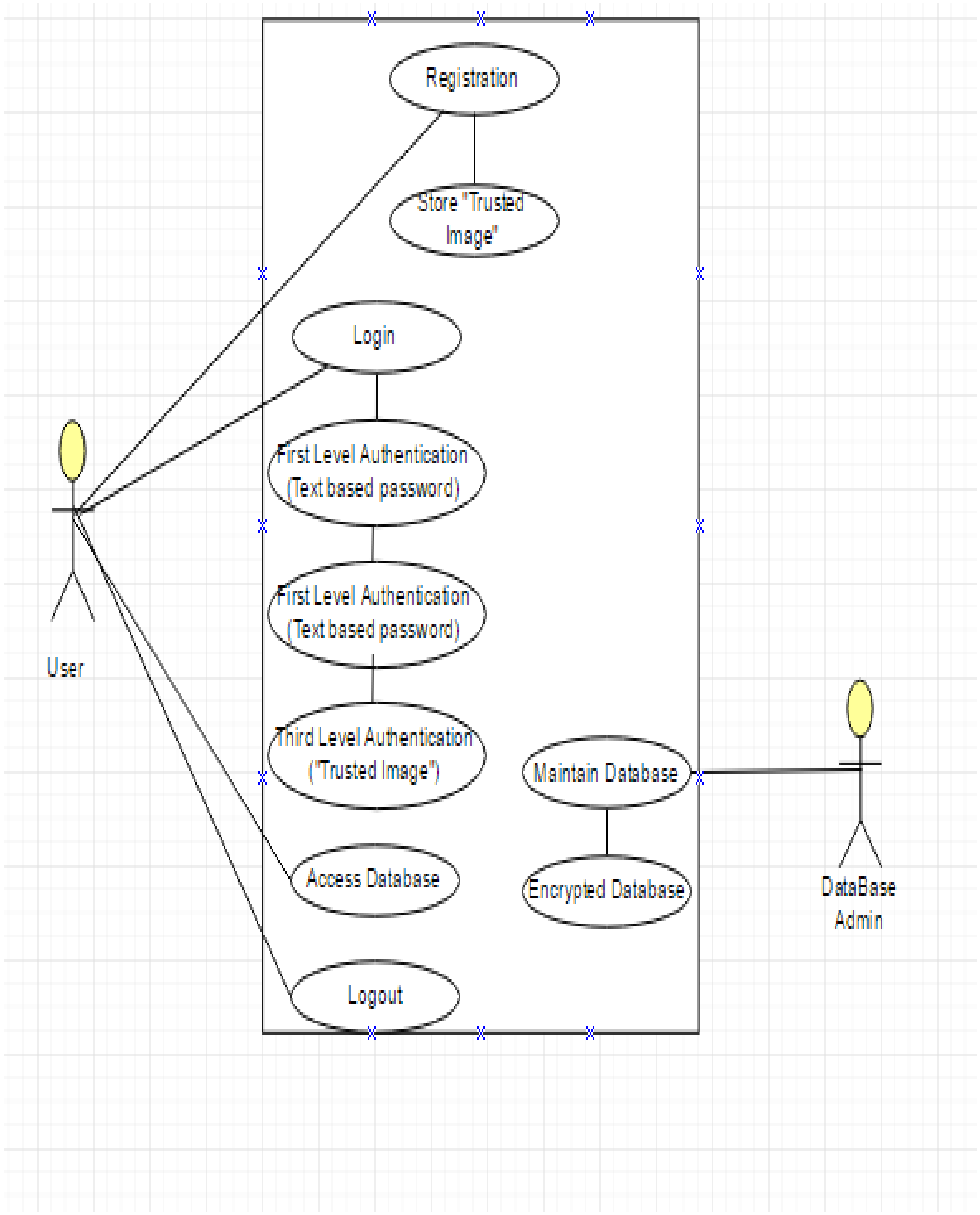
The third section of this report will detail the design and architecture of my system using UML diagrams, I will then discuss how the project will be implemented using snap shots of my code and any major algorithms to explain how my system works. I will also have some GUI layout detailing how the system will look from a user's points of view. Testing will then be carried out and described so that I can fully test the security of my system before presenting it. Expected response time and confidentiality, integrity and availability's will need to be tested.

In the fourth section, can then evaluate these tests and present a conclusion in the fifth section upon completion of the project.

The fifth section will then be further development plans for the project given time and resources.

Finally, I will present a bibliography of any sources I may have used that have contributed towards the final product that is my project as well as my monthly learning journals.

2 System


Registration
Store "Trusted Image"
Login
First Level Authentication (Text based password)
First Level Authentication (Text based password)
Third Level Authentication ("Trusted Image")
Access Database
Logout
Maintain Database
Encrypted Database
User
DataBase Admin

An **actor** in the Unified Modeling Language (UML) as shown above in this case a Use Case "specifies a role played by a user or any other system that interacts with the subject."

In the above use case I have defined two actors that will be interacting with my system, they are labelled “User” and “Database Administrator”. The “User” will be anyone who is accessing the system through the register and login functions, they must select a “Trusted Image” during registration and then login through the three levels of authentication detailed above. The “User will then be granted database access, based on their assigned roles and privileges they can access “CRUD” functionalities on the database data.

The second actor I have defined is the “Database Administrator”, this actor is responsible for maintaining the database integrity and ensuring that the correct roles are assigned to the correct users also that the hashing and encryption standard of the database is maintained to a high level.

3 Requirements

3.1 User Requirements Definition

- The end-user will be able to register an account.
- The end user will be able to login.
- The end-user will complete a first level text based password.
- The end-user will complete a second level multi-factor authentication.
- The end-user will complete an Advanced level image based authentication.
- The end-user will login successfully.
- The end-user will be presented with the database.
- The end-user user may modify the database.
- The end- user will log out of the system.

The end user will expect the system to adhere to the three security principles of confidentiality, integrity and availability.

The end user will expect the system to respond in a timely manner.

The end user will expect the system to be simple and intuitive to use.

3.2 Functional requirements

1. The system provides the user with the ability to register for an account.
2. The system stores the users selected image as a “trusted image”.
3. The system offers to log the user in to their account.
4. The system offers the first level of authentication the user in the form of a text based password.
5. The system offers the second level of authentication by requesting an e-mail or phone number to send an authentication code.
6. The system offers the Advanced level of authentication by requesting the user to provide their “trusted image”.
7. The system provides the user access to the database with Create, Read and Update functionality.
8. The system offers the user to continue working or to logout.

3.2.1 Functional requirements

Requirement 1

Registration

3.2.1.1 Description & Priority

The user must register their account before attempting to login. The user must enter a username, text based password for their first level of authentication a valid e-mail and password for their second level of authentication and select a "trusted image" as an Advanced level of authentication.

Severity: High

3.2.1.2 Use Case

Scope

The scope of this use case is to describe how the user must register their details to create a valid account to be used on the system.

Description

This use case describes the user registering their details to create a valid account to be used on the system.

Flow Description

Precondition

1. The user has accessed the web-service and clicked the register button.

2. The user is not already logged in with another account.

Activation

This use case starts when the user clicks on the register button.

Main flow

1.0 User clicks on the register button.

1.1 The system prompts the user to enter their user name in plain text.

1.2 The user inputs their username in plain text.

1.3 The system checks that the username has not been used before.

1.4 The system accepts the username as a valid username.

1.5 The system asks to enter a password in plain text.

1.6 The user enters their password in plain text.

1.7 The system checks that the password is at least 8 characters long, has capital letters, numbers and symbols present.

1.8 The system accepts the password as a valid password.

1.9 The system asks the user for a phone number and e-mail address.

2.0 The user enters their phone number and e-mail address.

2.1 The system sends a verification e-mail and text to the user.

2.2The user clicks the verify link.

2.3 The system is notified the e-mail and phone number have been verified.

2.4 The system asks the user to select an image to store as their “trusted image”.

2.5 The user selects an image to use as their “trusted image”.

2.8 The system saves the modified image to the database as a “trusted image”.

2.9 The next case will describe how the image is saved in the database.

3.0 The system returns the user to the login screen.

Alternate flow

A1 The user enters an invalid username or password.

A2 The user enters an invalid phone number or e-mail address.

A3 User already logged in with another account.

Exceptional flow

The user has not registered an account.

The user does not possess the correct credentials.

Termination

The system presents the second level of authentication to the user.

Post condition

The user must enter the second level of authentication in the form of a registered e-mail address of password.

Frequency of use

Every time the user wants to begin the login process.

Assumptions

The user wants to login to the system.

Notes and Issues:

This is an include and pre-condition for all other use cases.

Requirement 2

Store Trusted Image

Description & Priority

The image that the user selects during the registration process must be stored in an encrypted database to maintain its integrity and confidentiality.

Severity: High

3.2.1.3 Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope

The scope of this use case is to describe how

3.2.1.4 *Description & Priority*

This use case describes how the user selects an image to be registered to their account as a “trusted image”.

Flow Description

Precondition

1. The user has registered a first level of authentication in the form of a plain text password.
2. The user has registered a second level of authentication in the form of a multi-factor authentication by registering a phone number and an e-mail address to receive authentication code to.
3. The user is not already logged in with another account.

Activation

This use case starts when the has selected their image to register as their “trusted image”

Main flow

1.0 The system presents the user with a selection of images.

1.1 The user selects the image they want to have as their “trusted image”.

1.2 The system converts the metadata into hash codes and saves the hash codes to an encrypted database.

1.3 The system confirms that the image has been saved to an encrypted database.

1.4 The system registers the “trusted image” to the users account.

Alternate flow

A1 The user does not select an image.

A2 User already logged in with another account.

Exceptional flow

The trusted image does not register to the users registered account.

Termination

The system presents the user the login screen.

Post condition

The user now has a “trusted image” registered and can begin the login process.

Frequency of use

When the user sets up a “trust image” for their registered account.

Assumptions

The user wants to set up a trusted image.

Notes and Issues:

The user will be unable to complete registration or login without a stored “trusted image”.

Requirement 3

Login

3.2.1.5 Description & Priority

The user must enter their registered details user name, password, e-mail address/phone number and “trusted image” to login to the system successfully.

Severity: High

3.2.1.6 Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope

The scope of this use case is to describe how the user must enter their registered details in the form of a username, password, e-mail address/phone number and “trusted image” to login successfully.

Description

This use case describes the user logging in to the system with their registered in the form of a username, password, e-mail address/phone number and “trusted image”.

Flow Description

Precondition

1. The user has registered their details and successfully created an account.

2. The user is not already logged in with another account.

Activation

This use case starts when the user clicks the login button.

Main flow

1.0 User clicks on the login button.

1.1 The system prompts the user to enter their registered user name.

1.2 The user inputs their registered username.

1.3 The system checks that the username is registered.

1.4 The system accepts the username as a registered username.

1.5 The system asks the user to enter their registered password.

1.6 The user enters their registered password.

1.7 The system checks that the password is registered to the user’s account.

1.8 The system accepts the password the registered password.

1.9 The system asks the user to enter their registered phone number or e-mail address.

2.0 The user enters their registered phone number or e-mail address.

2.1 The system sends an authentication code to the user’s e-mail address or phone number

2.2The user enters the authentication code into the system.

2.3 The system accepts the authentication code.

2.4 The system asks the user to select an image.

2.5 The user selects an image.

2.6 The system here measures the integrity of the image by converting the new one into hash and comparing it against the “trusted image”.

2.7 The system authenticates the “trusted image”.

2.8 The user is logged in and can access the database.

Alternate flow

A1 The user enters a non-registered username or password.

A2 The user enters a non-registered phone number or e-mail address.

A3 The user enters the wrong “trusted image”.

A4 User already logged in with another account.

Exceptional flow

The user has not registered an account.

The user does not possess the correct credentials.

Termination

The system presents the user with the database.

Post condition

The user can now modify and delete data inside of the database.

Frequency of use

Every time the user wants to begin the login process.

Assumptions

The user wants to login to the system.

Notes and Issues:

This is an include and pre-condition for all other use cases.

Requirement 4

First level user authentication text based password.

Description & Priority

First level user authentication login. Users will be prompted to login with their first level credentials in the form of a text based password.

Severity: High

Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope

The scope of this use case is to describe how the user can log in to the first level of authentication using their text based password.

Description

This use case describes the user logging in to the first level of authentication using their text based password that they set up during the registration process.

Flow Description

Precondition

1. The user has a registered account.

2. The user is trying to log in with their registered account.

3. The user is not already logged in with another account.

Activation

This use case starts when the user has completed registration and attempts to login.

Main flow

1.0 User click on the login button.

1.1 The system prompts the user for their username and text based password.

1.2 The user enters their username and their text based password.

1.3 The system authenticates the username and password.

1.4 The user is then presented with the second level of authentication.

Alternate flow

A1 The user enters an invalid username or password.

A2 User already logged in with another account.

Exceptional flow

The user has not registered an account.

The user does not possess the correct credentials.

Termination

The system presents the second level of authentication to the user.

Post condition

The user must enter the second level of authentication in the form of a registered e-mail address of password.

Frequency of use

Every time the user wants to begin the login process.

Assumptions

The user wants to login to the system.

Notes and Issues: Extends login use case

Requirement 5

Second level user authentication login multi-factor authentication.

Description & Priority

Second level user authentication login in the form of a multi-factor authentication. Users will be prompted to send an authorization code to their registered phone or e-mail address.

Severity: High

Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope

The scope of this use case is to describe how the user can log in to the second level of authentication using a multi-factor authentication process.

Description

This use case describes the user logging in to the second level of authentication using their registered e-mail or phone number to receive an authentication code.

Flow Description

Precondition

1. The has passed the first level of authentication in the form of a text based password.

2. The user is trying to log in with their registered account.

3. The user is not already logged in with another account.

Activation

This use case starts when a user passes the first level of authentication with their registered text based password.

Main flow

1.1 The system prompts the user to enter their registered phone number or e-mail address.

1.2 The user enters their registered phone number/ e-mail address.

1.3 The system sends the authentication code to the user’s registered phone number/e-mail address.

1.4 The user receives their authentication code to their registered phone number/e-mail address.

1.5 The user enters the authentication code into the system.

1.6 The system checks the authentication code.

1.7 The user is presented with the Advanced level of authentication in the form of Image authentication.

Alternate flow

A1 The user enters an invalid phone number/ e-mail address.
A2 User already logged in with another account.

Exceptional flow

The user does not have knowledge of the registered phone or e-mail address.

Termination

The system presents the Advanced level of authentication to the user in the form of an image authentication.

Post condition

The user must enter the Advanced level of authentication using their stored “trusted image”.

Frequency of use

Every time the user passes the first level of authentication.

Assumptions

The user has passed the first level of authentication.

Notes and Issues:

This use case will only apply if the user can pass the first level of authentication.

Requirement 6

Advanced level user authentication login image level authentication.

Description & Priority

Advanced level user authentication login. Users will be prompted to login with their Advanced level credentials in the form of an image authentication, the user will use their selected image to check against the stored “trusted image” which was stored during the registration process.

Severity: High

Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope

The scope of this use case is to describe how the user can log in to the Advanced level of authentication using their stored “trusted image” which was stored during the registration process.

Description

This use case describes the user logging in to the Advanced level of authentication using their chosen registered “trusted image”.

Flow Description

Precondition

1. The user has passed the second level of authentication in the form of a multi-factor authentication.

2. The user is trying to log in with their registered account.

3. The user is not already logged in with another account.

Activation

This use case starts when a user passes the second level of authentication in the form of a multi-factor authentication.

Main flow

1.1 The system prompts the user to enter their “trusted image”.

1.2 The system compares the image provided with the stored “trusted image” that is registered to the users account.

1.3The system accepts the “trusted image”

1.4The user is granted access to the database.

Alternate flow

A2 The user has forgotten their “trusted image”.
A2 User already logged in with another account.

Exceptional flow

The user never set up a “trust image”.

Termination

The system presents the database to the user.

Post condition

The user can now modify the database.

Frequency of use

Every time the user passes the second level of authentication in the form of multi-factor authentication.

Assumptions

The user has passed the second level of authentication in the form of multi-factor authentication.

Notes and Issues:

This use case will only apply if the user can pass the second level of authentication in the form of multi-factor authentication.

Requirement 7

Access and modify Database with CRUD functionality’s

Description & Priority

Once the user has passed the three level of authentication they will be presented with a database that they can perform create, read, update and delete functionalities on.

Severity: High

Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope

The scope of this use case is to describe how the user can use CRUD functionalities on the database.

Description

This use case describes the user accessing the database after logging into all three authentication levels using their registered credentials.

Flow Description

Precondition

1. The user has passed the first level of authentication in the form of a text based password.

2. The user has passed the second level of authentication in the form of a multi-factor authentication using a registered email address/phone number to receive an authentication code.

3. The user has passed the Advanced level of authentication using their stored “trusted image”.

Activation

This use case starts when the user has passed all three level of authentication and has accessed the database.

Main flow

1.1 The system presents the database to the user.

1.2 The user can perform create, read, update and delete functions on the tables and given data in the database.

1.3The system asks the user to save their changes.

1.4The user saves their changes.

1.5The system accepts the changes.

Alternate flow

A1The user selects continue working.
A2 User already logged in with another account.

Exceptional flow

The system does not ask the user to logout.

Termination

The system accepts the changes.

Post condition

The user is presented with the login screen.

Frequency of use

Whenever the user has passed all three levels of the authentication and wants to create, read, update or delete anything from the database.

Assumptions

The user wants to make changes to the contents of the database.

Notes and Issues:

The user will not always want to modify the database.

Requirement 8

Logout

1.1.1.1 Description & Priority

Once the user has made their desired changes to the database the system will then offer the option to logout or to continue working.

Severity: Medium

1.1.1.2 Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope

The scope of this use case is to show how the system will offer the user to log out or to continue working.

Description

This use case describes the user being offered to logout of the database or to continue working on the data.

Flow Description

Precondition

1. The user has passed the first level of authentication in the form of a text based password.

2. The user has passed the second level of authentication in the form of a multi-factor authentication using a registered email address/phone number to receive an authentication code.

3. The user has passed the Advanced level of authentication using their stored “trusted image”.

4. The user has performed CRUD functionalities on the database.

5. The user has saved their changes to the database.

Activation

This use case starts when a has passed all three level of authentication and has accessed the database and modified the database.

Main flow

1.1 The system asks the user if they want to logout or if they want to continue working.

1.2 The user selects logout.

1.3 The user is returned to the login screen

2.0 The user selects continue working.

2.1The system allows the user to remain in the database with the same CRUD functionalities.

Alternate flow

A1 The user does not select logout or continue working.

Exceptional flow

The system crashes.

Termination

The system places the user back at the login screen

Post condition

The user can decide to login again or to remain logged out of the system

Frequency of use

Whenever the user has committed their changes to the database.

Assumptions

The user has committed changes to the database.

Notes and Issues:

The user will not always want to modify the database

4 Non-Functional Requirements

4.1 Data requirements

In this section I will describe the various data types which will be essential for implementing the functionality of my system.

Azure SQL Database: This database is hosted on the cloud through azure, I can make changes to the database by using the “Database Explorer” found in visual studio, this will allow me to set user permissions, check who is a registered user and check details of other users. Another table will hold the details of my “Sensitive data” for my project. Data will be stored through various methods in controllers that allow me to generate the local DB files.

The “trusted image” will be contained in another table in the Azure SQL Database.

All other data will be held as strings and numbers that have been converted to hash files using a 256 AES Encryption Method.

Users will input in plain text, numbers and symbols.

The Image reader can only read certain files to prevent malicious file upload, the whitelisted file extensions are .jpg, .bmp, .gif, .png.

4.1.1 User requirements

In this section, I will talk about the user requirements. These are essential requirements that the user must have to use the application.

- The user will need access to a computer / smart phone capable of browsing the internet.
- The user will need to know the URL of the web application.
- The user will need to register an account
- The user will need to login
- The user will need to have a permission granted to make modifications to the database.

- The user must logout or be timed out.

4.1.2 Environmental requirements

In this section, I will talk about the environmental requirements. These are the vital requirements that must be present when developing the application.

- The system will need to be run on a windows machine from windows 7 onwards.
- The system will be developed in visual studio.
- The database and web application will be hosted on Microsoft azure cloud.
- The application can only be run on a https URL for security.
- The system will need internet access so that I can push and commit to
- GitHub and connect to the database on the cloud and also publish the application for testing and to make changes.

4.1.3 Usability requirements

Understandability: The interface should be easy to understand and navigate.

Operable: The system should be consistent and available always. Any down time should be explained, error messages should clearly point to the source of the problem.

Integrity: The system should ensure that no data is manipulated unless permission has been granted for the data to be manipulated. The data should be consistent and correct at all times.

Confidentiality: The system should keep both user and sensitive data confidential, only those who have the correct permissions to view the data should be able to.

Learnability: The system should be self-explanatory and easy to learn.

Performance/Response time requirement

The system should have a minimal response time of one second or less when navigating through the login, authentication and database. The logout may take slightly longer as to securely log the user out this time can exceed no longer than two seconds. Performance wise the system should be able to handle an average number of daily users with ease, as it is a security application and will need to have little to no down time the system response and uptime will be of high concern.

Availability requirement

The system will need to be available 24 hours a day, if any maintenance needs to take place the users will be informed and the maintenance will be performed during off peak hours primarily between 00.00 – 01.00 am where applicable.

Recover requirement

I will be using a full recovery model for my SQL server database. The system will require log backups. This will ensure that no work is lost due to a lost or damaged data file. The system can recover to an arbitrary point in time (for example, prior to application or user error). Work loss exposure is usually none also if the tail of the log is damaged, changes since the most recent log backup must be redone. The system can recover to a specific point in time, assuming that the backups are complete up to that point in time.

Robustness requirement

The listener shall not terminate in case of invalid API-requests; the listener shall reject those.

Threat: The listener terminates in case of an invalid API-request.

Design Pattern: Strong input validation

Testing this requirement: Call API with wrong number of parameters, with wrong data types, with values exceeding the allowed range, with special characters e.g. white spaces, control-sequences, characters.

The listener shall be resistant against unexpected flood of requests (which could be a real unexpected load-peak, the result of errors in the calling application resulting in infinite number of retries, or an intentional attack).

Threat: The listener terminates or stalls under an unexpected flood of requests.

Design Pattern: Ability to count and limit the number of requests or open sessions accepted per source (server, IP-number), username. Return a defined error message.

Testing: Submit more requests then the API can handle.

Comment: The calling application needs to treat this error message accordingly!

The database shall be protected from reaching the session limit caused by unexpected number of database connections, e.g. from application processes or interfaces opening a new database connection per request without closing / log out previous database session.

Threat: The database reaches the session limit and rejects new sessions resulting in a partial unavailability of service.

Design Pattern: Use either "only one database session per application process instance" or use connection / session pooling.

Testing this requirement: Verify that the number of database sessions does not increase beyond the required and designed level.

Monitoring: monitor number of database sessions and warn / alert when reaching warning and alarm threshold.

Last line of defence: Locking database account of that interface, user or application module causing too many database sessions.

Pre-Requisite: using different user accounts for each interface and application module.

Security requirement

The system will offer first, second and Advanced level authentication and adhere to the security principles of Integrity, Availability and Confidentiality.

The application that the project is based on as well as all the security features will be written using secure coding methods.

The first level of authentication will be in the form of a text based password, registered to the users account the password must be at least 8 characters long, contain numbers and letter both uppercase and lowercase and contain at least one special character, this password will be stored in an encrypted database using AES 256 Encryption Method.

256-bit encryption refers to the length of the encryption key used to encrypt the sensitive data. A hacker will require 2256 different combinations to break a 256-bit encrypted message this is virtually impossible to be broken by even the fastest computers.

Typically, 256-bit encryption is used for data in transit, or data traveling over a network or Internet connection. However, in my system it will be implemented for sensitive and important data. The US government has adopted 256-bit encryption to protect classified and top secret communication and files so I have faith it will be able to protect any data that will be protected by my system.

The second level of authentication will be in the form of a multi-factor authentication, the user will need to register an e-mail address and phone number to their account, the user will receive a onetime authentication code to their chosen device allowing them to login successfully these user details will be stored in an encrypted database using AES 256 Encryption Method.

The Advanced level of authentication will be in the form of image level authentication the user will register select an image during registration the images metadata will be saved the image will then be converted to hash files, the now “trusted image” will be stored in an encrypted database using AES 256 Encryption Method.

The link between the web server and the browser will be protected by SSL ensuring that all data passed between the web server and browsers remain private and integral.

The database will have user restrictions and access controls imposed upon the database allowing only users with the correct credentials to access the appropriate functionalities.

My system will also run a log of all actions carried out on the database the system will log all administrator activity. Log the deletion of any data. Log any modification to data characteristics: permissions, location and field type.

All code that accepts input from users via an HTTP request will be reviewed to ensure that it can identify large input. Once inappropriate data is identified the activity will be logged and the data dropped. All data input fields will have reasonable field lengths and specific data types. This can be done by limiting the amount of text allowed in free form fields.

Reliability requirement

The system should be reliable in the form that the AES 256 Encryption has yet to be cracked, even with a supercomputer, it would take **1 billion years** to crack the 128-bit AES key using brute force attack. This is more than the age of the universe (13.75 billion years). So, given that the AES 256 is more than twice as difficult it is safe to say the encryption will be as secure as possible.

The SSL will allow all traffic between the server and browser to remain encrypted and protected from interception or “man in the middle” attacks.

As the database will be fully backed up with a full recovery model, file backup will be consistent and work loss will not happen.

Maintainability requirement

The system will be easy to maintain and modify as long as the user has the correct access privileges and rights. Maintainability will only be necessary for modifications that are made between releases of major new versions of the application or component. These will include minor defect and enchantments throughout the systems lifetime.

The database will be maintained at the same time as the system, the database will constantly be logged and backed up. If any critical corruption or failures occur, we will be able to roll back or restore the database with minimal work loss.

Downtime expected from any form of maintenance should be less than 15 minutes and will be performed at off peak hours where possible to best accommodate the user.

Portability requirement

The system will be deployed both as a web application and will also be available for a mobile environment. The system will be deployable on as many devices as possible and as many environments as possible. The system will be primarily developed in C# and so should be relatively easy to move from software environment to the next.

Reusability requirement

As the system is going to be deployable across multiple platforms and software environments, a minimum of 30% of the application's software shall be potentially reusable on future endeavours. I will focus on making as much of this re-usable as possible. This will become clearer at a later point in the project development.

Interface requirements

User Interfaces: The Interface of the user side web application will have a register button and a log in button on the home page. The user will first have to click on the register button, the system will then present them with the several text fields to fill out to register their account details. The user will then input their data into these 5 text fields. The last field will be used to upload an image using a "browse" button. The user can then press the "submit" button. This will submit the modified image and the 5 prior text fields. This will provide the user with a registered account with a username, password, multi-factor authentication and a "trusted image".

The user will then return to the home page and select the “login” button. The user will then be presented with two text fields one asking for the registered username and the other asking for their registered password the user can then click the “submit” button. Once this is clicked the system checks the details against the database if successful the user will be presented with another text field asking the user to enter their registered e-mail address or phone number. The user enters their details and the system send the user a text or e-mail, the user is the presented with a blank text field asking for their authorization code.

Once the authorization code is verified, the user will be presented with a “Browse” button, a “Submit” button and a box to display an image. Once the user has clicked the “Browse” button they can select from a selection of images in a folder. The user will then find their “trusted image” in the folder once the “trusted image is found the user will upload the file the user can then see the “trusted image” the user can then click on “submit” or “browse”. If the user presses browse they return to the folder of images, if the user selects submit the image will be checked against the stored “trusted image” in the database. If the image matches the user will be sent to the Web page representing the database data. If the image does not match the “trusted image” the user will be warned via red text appearing in an alert box warning that they have 1 more attempt before being locked out.

Hardware Interfaces: The hardware I will be using during the development of my system will be the college computers, some run on Citrix which is a virtual machine while others run on the local machine. I will be doing the majority of my work on the local machines when available as the virtual environment seems to throw up difficulties when communicating with servers which will be essential as I am creating a live web application. I will also be making my web application available to mobile devise so it will not be a relatively light system resource wise and will not need a large amount of committed memory or high ram to be run smoothly. The information will not write information directly to the user’s computer but will instead be using a database which will be located on a networks server.

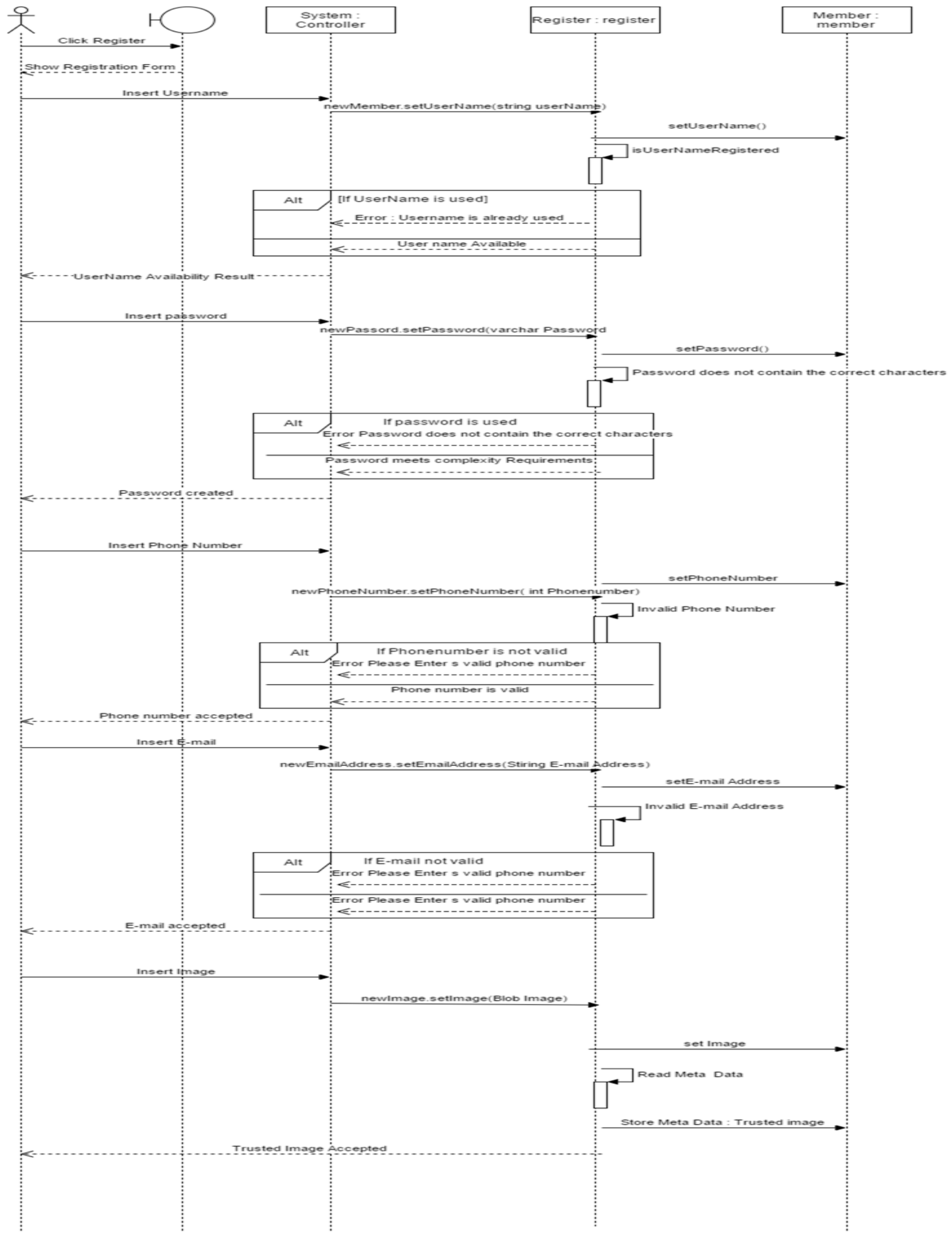
Software Interfaces: The system requires a properly configured version of windows 7 or windows 10. The computer must have Microsoft.net framework. The computer must also have a properly configured version of MySQL installed and configured.

Communication Interfaces: All data transferred between the server and the individual computer will be encrypted using an SSL certificate to ensure the data cannot be intercepted and only the intended recipient receives the data.

Application Programming Interfaces (API)

My system will use visual studios to provide the bulk of API'S needed for the system to work correctly, the data will be sent to and from a database created in visual studio using the .net framework, the data will be contained in tables written in MySQL and the messages will be transmitted between the website and database over an SSL certified transfer channel ensuring complete and total end to end encryption of all data.

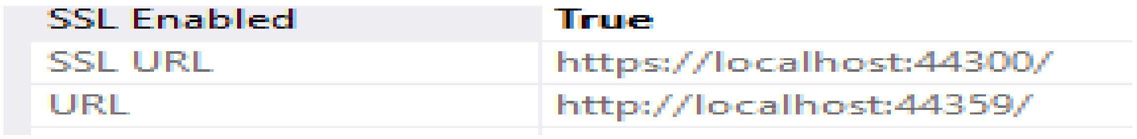
5 Design and Architecture (Registration Sequence Diagram)


System : Controller
Register : register
Member : member
Click Register
Show Registration Form
Insert Username
newMember.setUserName(string userName)
setUserName()
isUserNameRegistered
Alt
[If UserName is used]
Error : Username is already used
User name Available
UserName Availability Result
Insert password
newPassord.setPassword(varchar Password
setPassword()
Password does not contain the correct characters
Alt
If password is used
Error Password does not contain the correct characters
Password meets complexity Requirements
Password created
Insert Phone Number
setPhoneNumber
newPhoneNumber.setPhoneNumber(int Phonenumber)
Invalid Phone Number
Alt
If Phonenumber is not valid
Error Please Enter s valid phone number
Phone number is valid
Phone number accepted
Insert E-mail
newEmailAddress.setEmailAddress(Stiring E-mail Address)
setE-mail Address
Invalid E-mail Address
Alt
If E-mail not valid
Error Please Enter s valid phone number
Error Please Enter s valid phone number
E-mail accepted
Insert Image
newImage.setImage(Blob Image)
set Image
Read Meta Data
Store Meta Data : Trusted image
Trusted Image Accepted

My web application has been built using primarily C# in the visual studio MVC templated environment, my project works using controllers and migrations to handle the information transfer between the web application and the SQL database hosted on the cloud, this cloud database holds the user registration information to make sure only registered users can enter the database and make modifications, this can be seen from the sequence diagram above. The data is presented with a view of the registration form they then enter the required fields while the credentials are checked against requirements I have set up to ensure all methods of access are secure, valid and unique to each user wishing to register on the system.

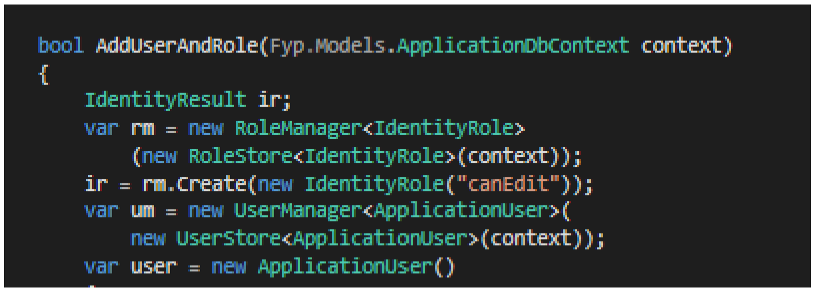
5.1 Implementation

As this is a security project, the first thing to do was to address the matter of setting up a HTTPS domain for my project to run live on, this can be done easily within visual studio MVC from changing the SSL settings to true. The SSL will ensure that I have an encrypted link between the web browser and the server, data passing between the browser and server will remain private and integral.



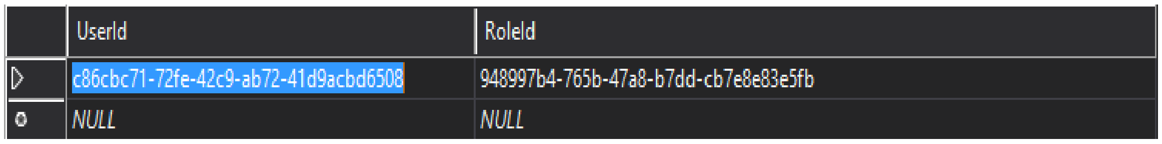
SSL Enabled	True
SSL URL	https://localhost:44300/
URL	http://localhost:44359/

I have also implemented access control using user roles within the database so that once a user has been logged in if they are on the Access List they will be granted a role to edit the data held in the data base.



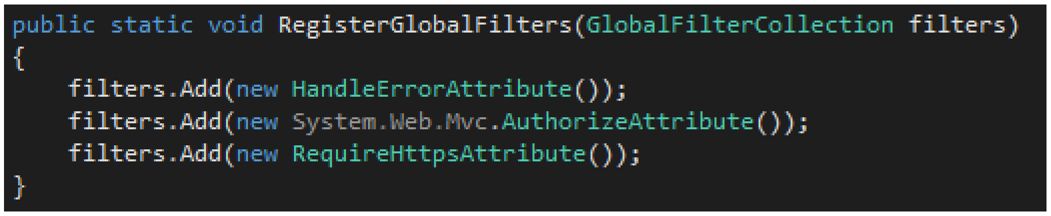
```
bool AddUserAndRole(Fyp.Models.ApplicationDbContext context)
{
    IdentityResult ir;
    var rm = new RoleManager<IdentityRole>
        (new RoleStore<IdentityRole>(context));
    ir = rm.Create(new IdentityRole("canEdit"));
    var um = new UserManager<ApplicationUser>(
        new UserStore<ApplicationUser>(context));
    var user = new ApplicationUser()
```

The user role and id is then stored in the SQL database in an encrypted format



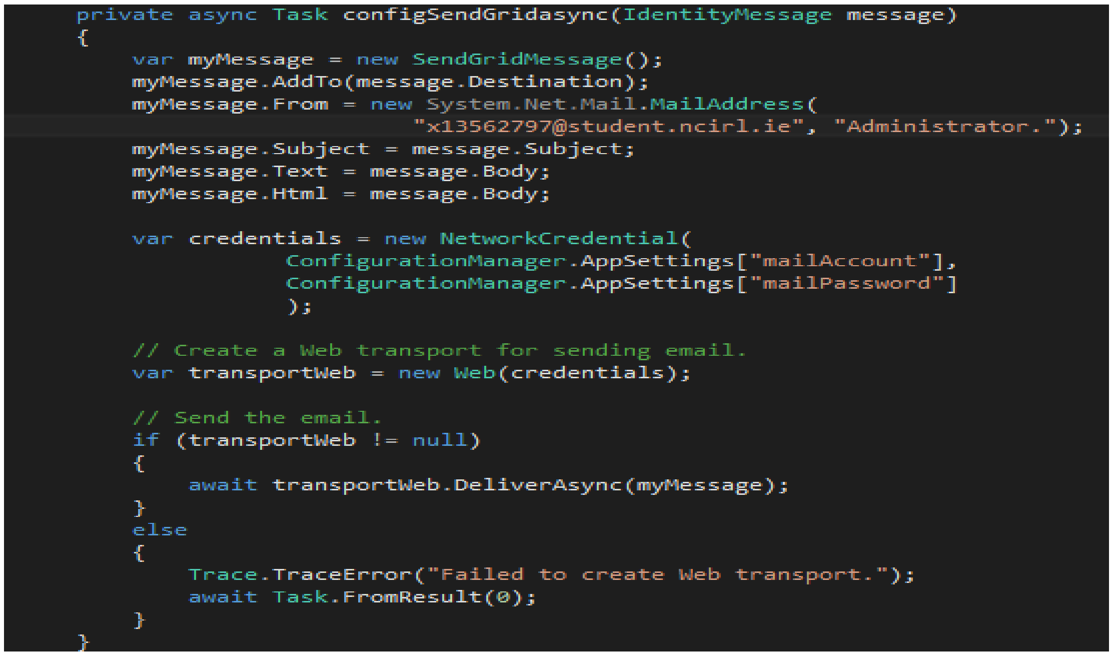
	UserId	RoleId
▷	c86cbc71-72fe-42c9-ab72-41d9acbd6508	948997b4-765b-47a8-b7dd-cb7e8e83e5fb
●	NULL	NULL

This code adds the Authorize filter and the RequireHttps filter to the application. The Authorize filter prevents anonymous users from accessing any methods in the application. The RequireHttps requires that all access to the web app be through HTTPS.



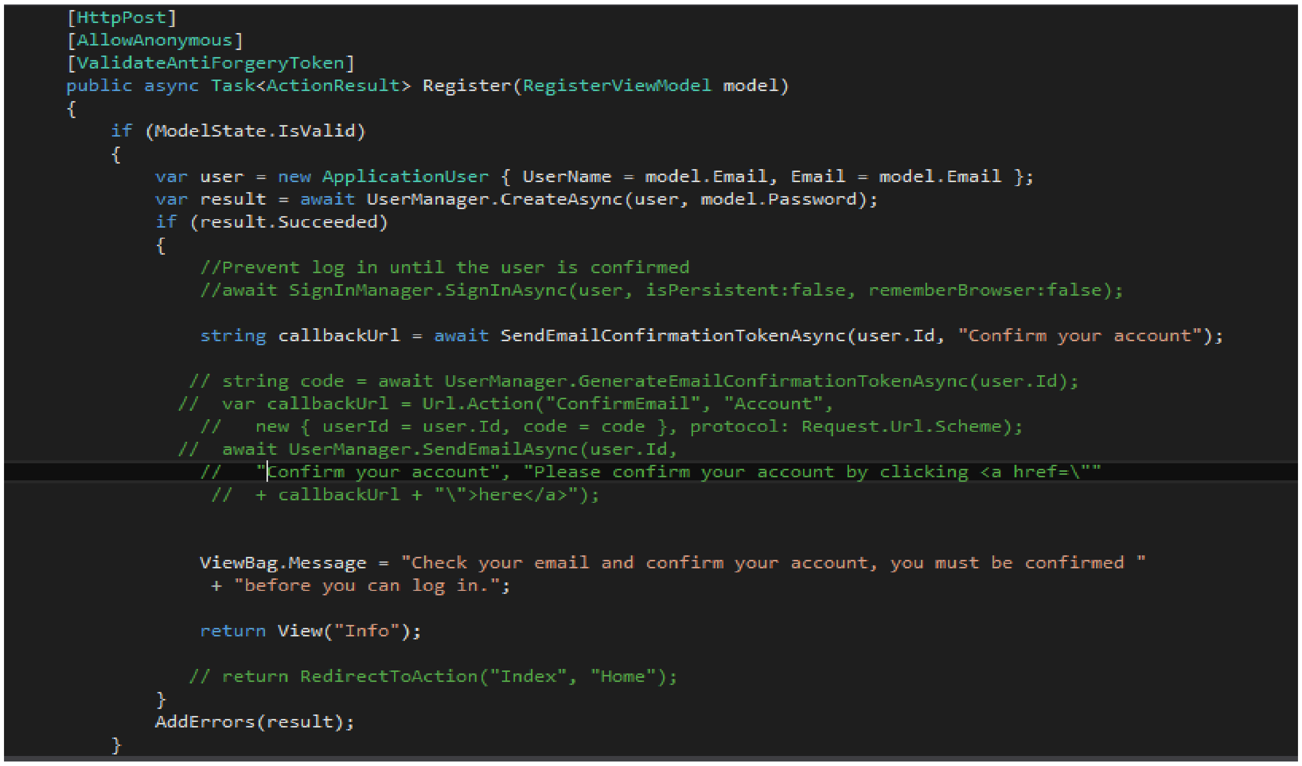
```
public static void RegisterGlobalFilters(GlobalFilterCollection filters)
{
    filters.Add(new HandleErrorAttribute());
    filters.Add(new System.Web.Mvc.AuthorizeAttribute());
    filters.Add(new RequireHttpsAttribute());
}
```

The code segment below shows how I have hooked up Send Grid an external application to handle E-mail Confirmation and Password Resets. This code basically shows the name of the E-mail account that will send the confirmation e-mail “Administrator”, using my network credentials that I have set up on the application marked as “mailAccount/mailPassword”. The web transport is then created and the e-mail is sent.



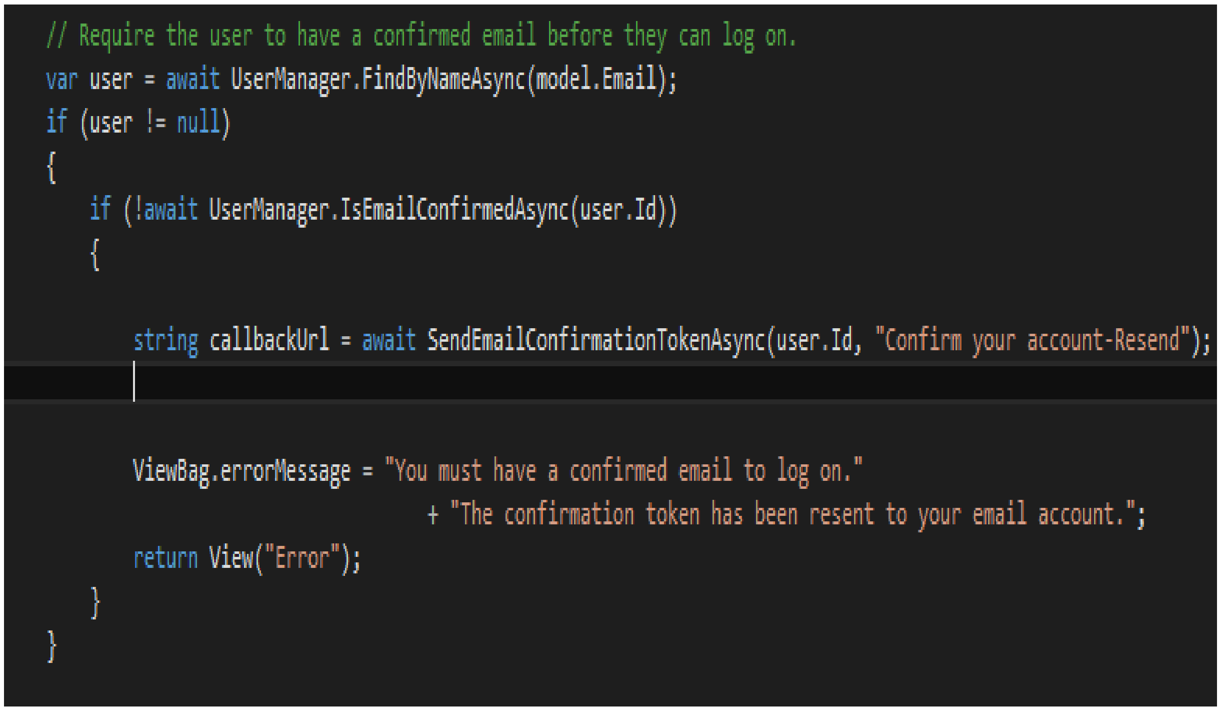
```
private async Task configSendGridasync(IdentityMessage message)
{
    var myMessage = new SendGridMessage();
    myMessage.AddTo(message.Destination);
    myMessage.From = new System.Net.Mail.MailAddress(
                        "x13562797@student.ncirl.ie", "Administrator.");
    myMessage.Subject = message.Subject;
    myMessage.Text = message.Body;
    myMessage.Html = message.Body;

    var credentials = new NetworkCredential(
                ConfigurationManager.AppSettings["mailAccount"],
                ConfigurationManager.AppSettings["mailPassword"]
                );

    // Create a Web transport for sending email.
    var transportWeb = new Web(credentials);

    // Send the email.
    if (transportWeb != null)
    {
        await transportWeb.DeliverAsync(myMessage);
    }
    else
    {
        Trace.TraceError("Failed to create Web transport.");
        await Task.FromResult(0);
    }
}
```

The commented “SignInAsync” Method allowed the user to login without confirming their e-mail, I have since removed this line to prevent the user logging into my project until the user has been confirmed. Although this will not completely ensure protection from bots and spam campaigns, it is at least some deterrent.



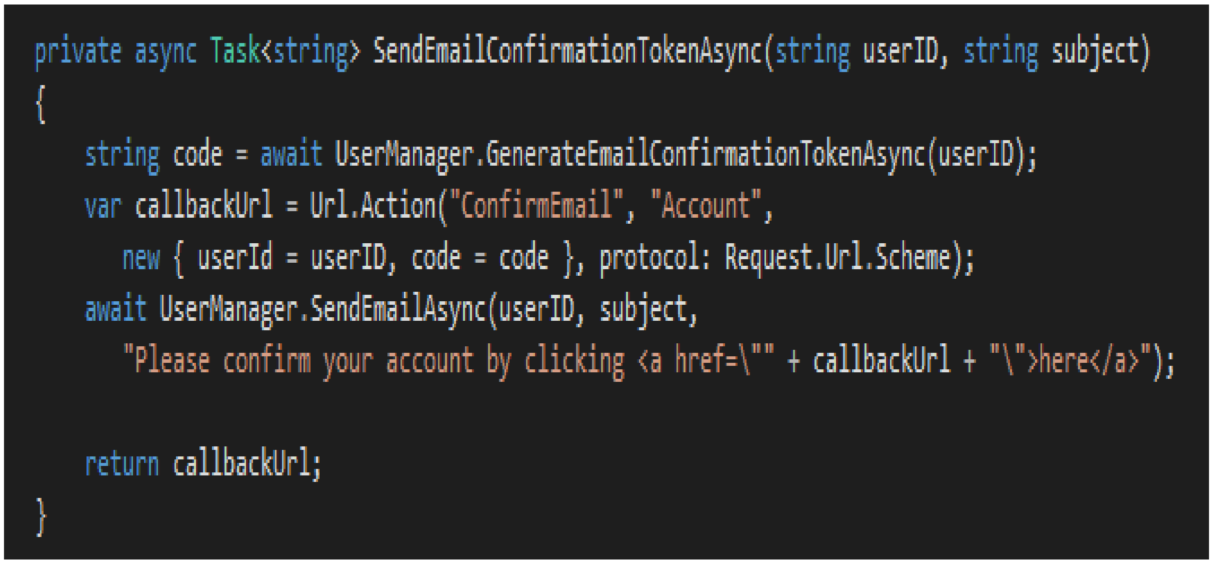
```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser { UserName = model.Email, Email = model.Email };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            //Prevent log in until the user is confirmed
            //await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false);

            string callbackUrl = await SendEmailConfirmationTokenAsync(user.Id, "Confirm your account");

           // string code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
          //  var callbackUrl = Url.Action("ConfirmEmail", "Account",
            //   new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);
          //  await UserManager.SendEmailAsync(user.Id,
            //   "Confirm your account", "Please confirm your account by clicking <a href=\""
             //  + callbackUrl + "\">here</a>");


            ViewBag.Message = "Check your email and confirm your account, you must be confirmed "
             + "before you can log in.";

            return View("Info");

           // return RedirectToAction("Index", "Home");
        }
        AddErrors(result);
    }
```

This section of code extends the previous one, it reminds the user to confirm their e-mail before they can successfully login. If they does not receive the e-mail or deletes the e-mail I have added a line of code to re-send the e-mail to the same e-mail address they are trying to register.



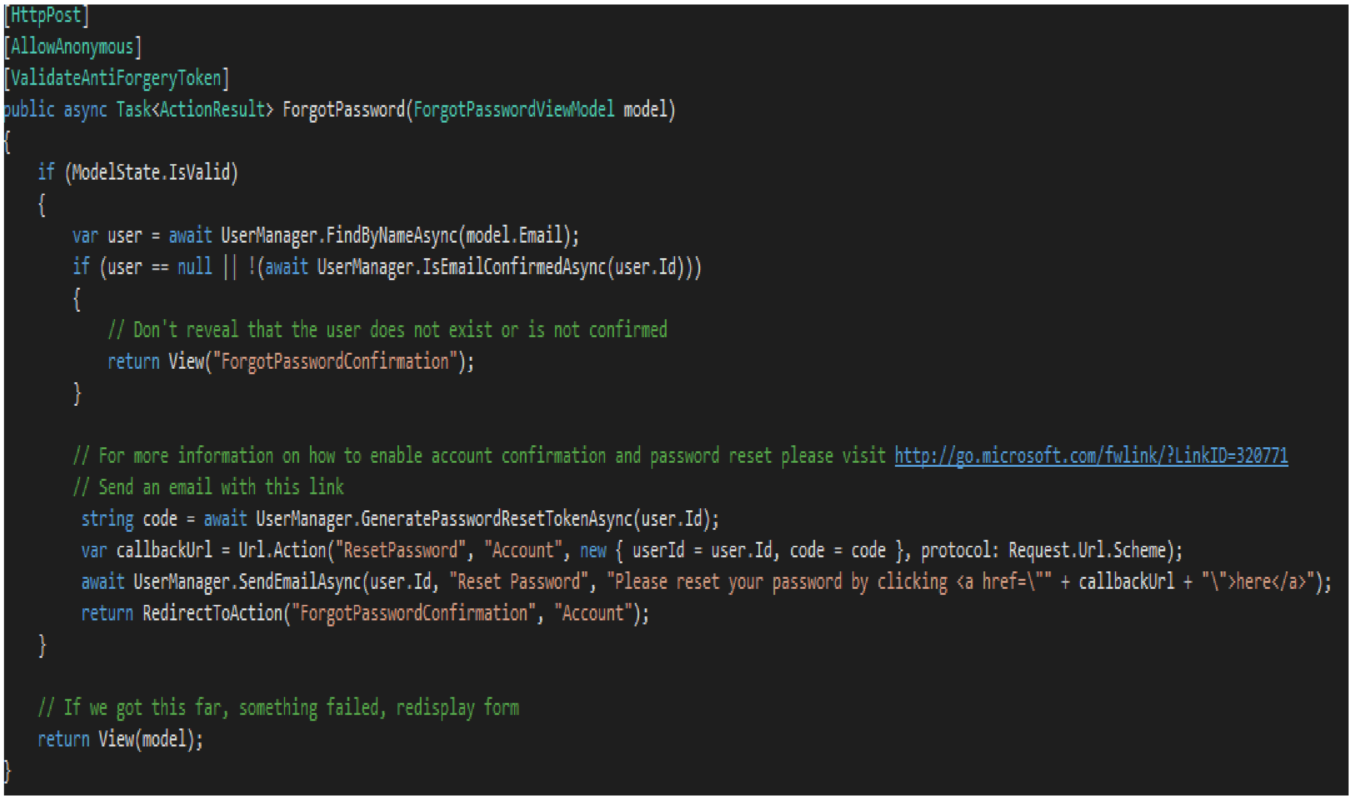
```
// Require the user to have a confirmed email before they can log on.
var user = await UserManager.FindByNameAsync(model.Email);
if (user != null)
{
    if (!await UserManager.IsEmailConfirmedAsync(user.Id))
    {

        string callbackUrl = await SendEmailConfirmationTokenAsync(user.Id, "Confirm your account-Resend");


        ViewBag.errorMessage = "You must have a confirmed email to log on."
                             + "The confirmation token has been resent to your email account.";
        return View("Error");
    }
}
```

This is the help method that allows the above line of code to send a replacement e-mail to the user attempting to register.



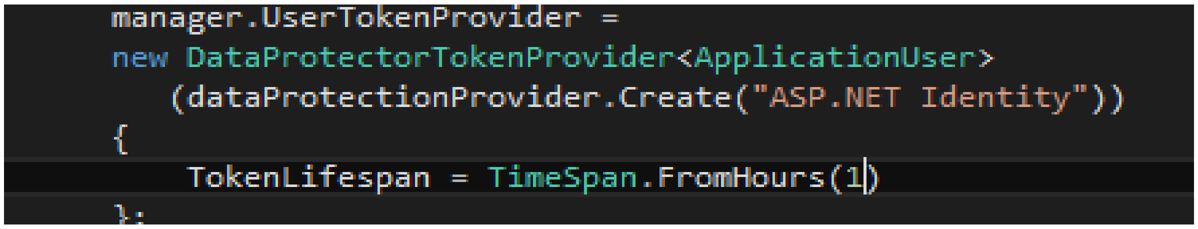
```
private async Task<string> SendEmailConfirmationTokenAsync(string userID, string subject)
{
    string code = await UserManager.GenerateEmailConfirmationTokenAsync(userID);
    var callbackUrl = Url.Action("ConfirmEmail", "Account",
       new { userId = userID, code = code }, protocol: Request.Url.Scheme);
    await UserManager.SendEmailAsync(userID, subject,
       "Please confirm your account by clicking <a href=\"" + callbackUrl + "\">here</a>");

    return callbackUrl;
}
```

This Section of code allows the password to be reset first it checks to see if the user is attached to the registered e-mail list, if the user is not registered by e-mail then they cannot be sent a password reset as the account has yet to be confirmed. The post method will send an email with a link to reset the password within the web application.



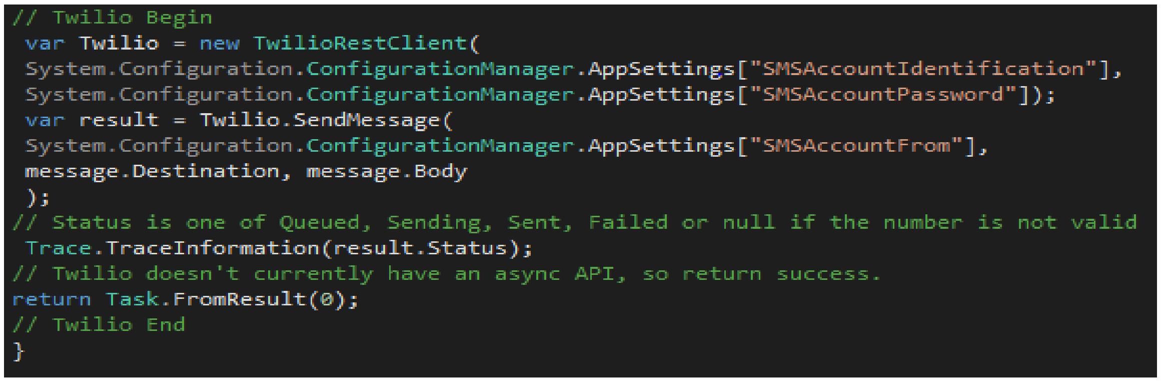
```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> ForgotPassword(ForgotPasswordViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = await UserManager.FindByNameAsync(model.Email);
        if (user == null || !(await UserManager.IsEmailConfirmedAsync(user.Id)))
        {
            // Don't reveal that the user does not exist or is not confirmed
            return View("ForgotPasswordConfirmation");
        }

        // For more information on how to enable account confirmation and password reset please visit http://go.microsoft.com/fwlink/?LinkID=320771
        // Send an email with this link
         string code = await UserManager.GeneratePasswordResetTokenAsync(user.Id);
         var callbackUrl = Url.Action("ResetPassword", "Account", new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);
         await UserManager.SendEmailAsync(user.Id, "Reset Password", "Please reset your password by clicking <a href=\"" + callbackUrl + "\">here</a>");
         return RedirectToAction("ForgotPasswordConfirmation", "Account");
    }

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

The method fails silently if the user email has not been confirmed. If an error was posted for an invalid email address, malicious users could use that information to find valid userId (email aliases) to attack. To prevent this I have used the following code to ensure all token expire even if they have not been used after 1 hour from the original time they were sent this will apply to the e-mail confirmation token, the 2FA token and the password reset token.



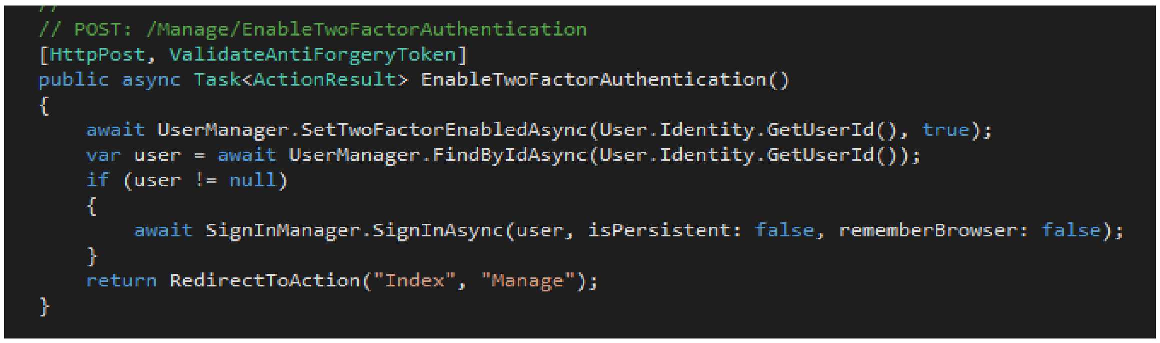
```
manager.UserTokenProvider =
new DataProtectorTokenProvider<ApplicationUser>
   (dataProtectionProvider.Create("ASP.NET Identity"))
{
    TokenLifespan = TimeSpan.FromHours(1)
};
```

Similar to setting up the e-mail confirmation and password reset, my multi-factor authentication uses an external API to send text messages containing verification codes to the user as a second layer of security once the user has logged in past the confirmation e-mail stage.



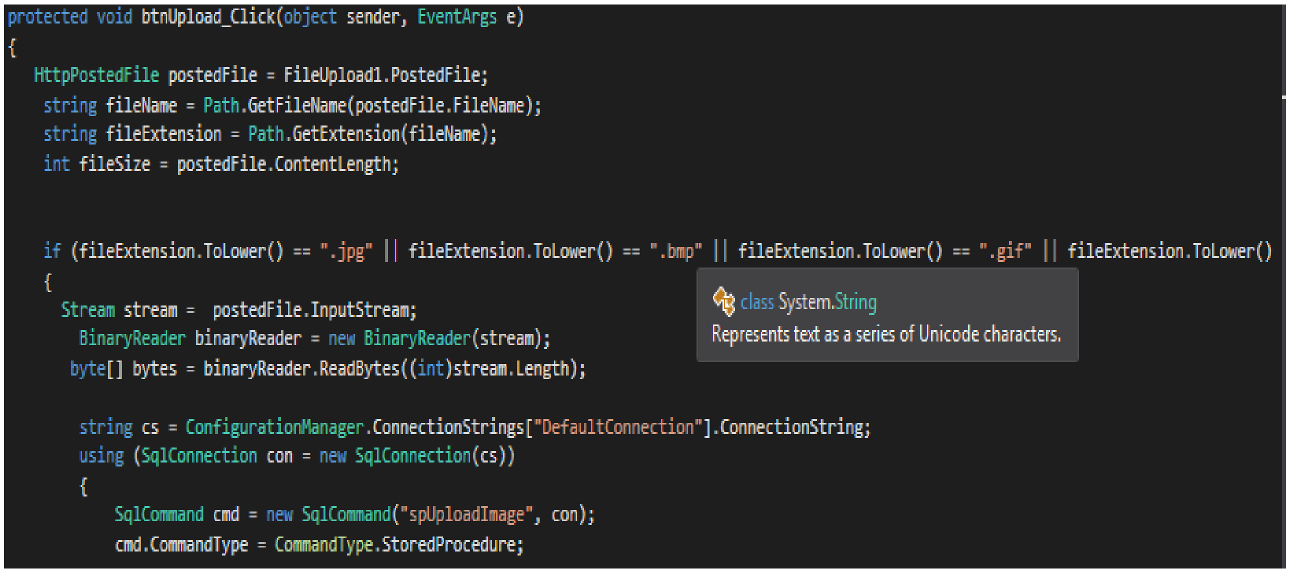
```
// Twilio Begin
 var Twilio = new TwilioRestClient(
 System.Configuration.ConfigurationManager.AppSettings["SMSAccountIdentification"],
 System.Configuration.ConfigurationManager.AppSettings["SMSAccountPassword"]);
 var result = Twilio.SendMessage(
 System.Configuration.ConfigurationManager.AppSettings["SMSAccountFrom"],
 message.Destination, message.Body
 );
// Status is one of Queued, Sending, Sent, Failed or null if the number is not valid
 Trace.TraceInformation(result.Status);
// Twilio doesn't currently have an async API, so return success.
return Task.FromResult(0);
// Twilio End
}
```

The below method enables to Two factor authentication to be run, I have highlighted the “ValidateAntiForgeryToken” This attribute prevents external get request and cross site scripting ensuring no attackers can use this method. Forgery tokens are only generated once per session.

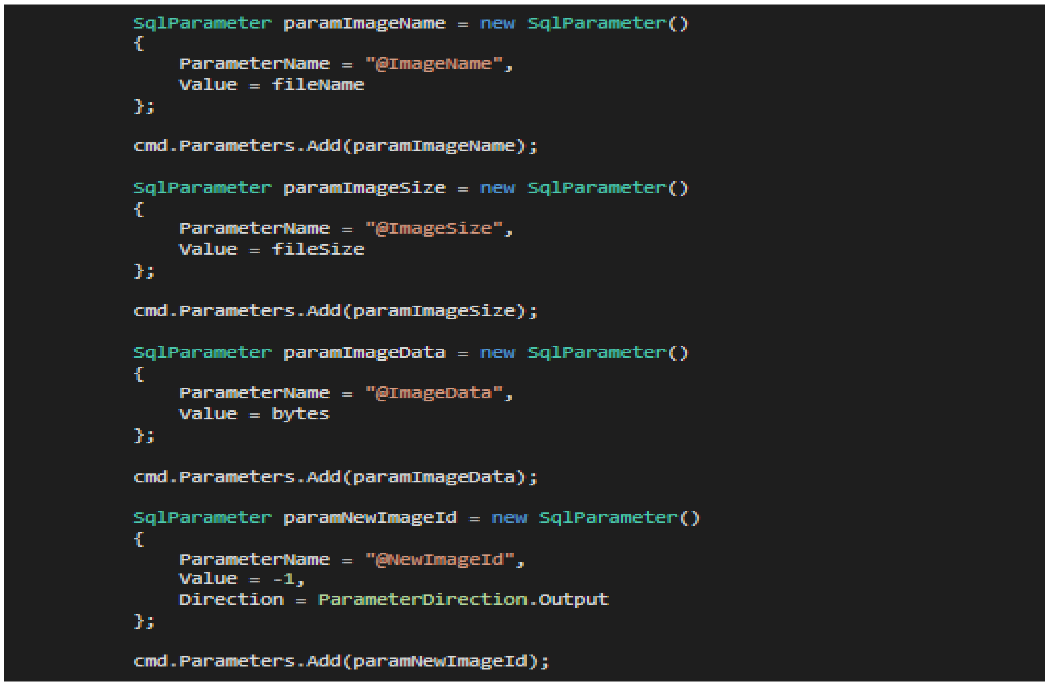


```
// POST: /Manage/EnableTwoFactorAuthentication
[HttpPost, ValidateAntiForgeryToken]
public async Task<ActionResult> EnableTwoFactorAuthentication()
{
    await UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), true);
    var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
    if (user != null)
    {
        await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
    }
    return RedirectToAction("Index", "Manage");
}
```

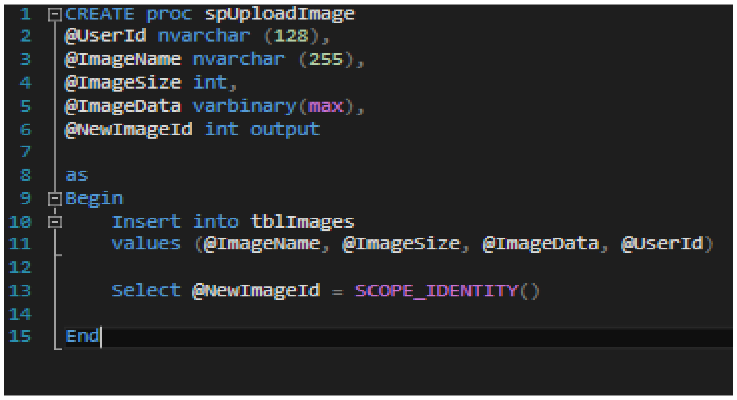
Below you can see the code for my file image reader, this code reads the various file extension name that my system is to accept if the picture that is being uploaded does not match they will receive an error this will prevent malicious file upload. The code also implements a binary reader so that we can convert the image into byte to allow us to store the image in the database, the file reader is connected to my database use “Default Connection” and we use the Stored Procedure I created to read the image into the database using the SQL parameters from the stored procedure in the database.



```
protected void btnUpload_Click(object sender, EventArgs e)
{
   HttpPostedFile postedFile = FileUpload1.PostedFile;
    string fileName = Path.GetFileName(postedFile.FileName);
    string fileExtension = Path.GetExtension(fileName);
    int fileSize = postedFile.ContentLength;


    if (fileExtension.ToLower() == ".jpg" || fileExtension.ToLower() == ".bmp" || fileExtension.ToLower() == ".gif" || fileExtension.ToLower()
    {
      Stream stream =  postedFile.InputStream;
        BinaryReader binaryReader = new BinaryReader(stream);
       byte[] bytes = binaryReader.ReadBytes((int)stream.Length);

        string cs = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;
        using (SqlConnection con = new SqlConnection(cs))
        {
            SqlCommand cmd = new SqlCommand("spUploadImage", con);
            cmd.CommandType = CommandType.StoredProcedure;
```

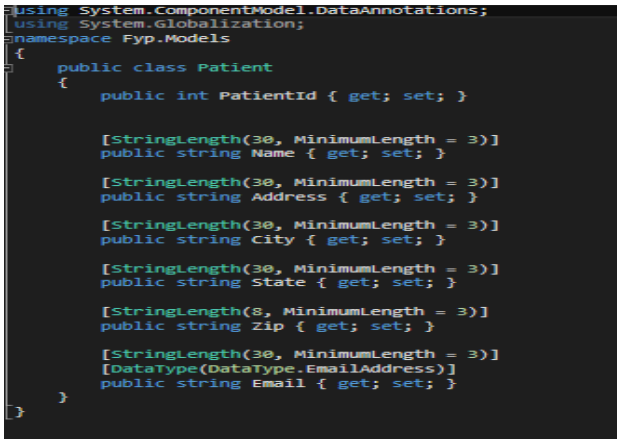


```
            SqlParameter paramImageName = new SqlParameter()
            {
                ParameterName = "@ImageName",
                Value = fileName
            };

            cmd.Parameters.Add(paramImageName);

            SqlParameter paramImageSize = new SqlParameter()
            {
                ParameterName = "@ImageSize",
                Value = fileSize
            };

            cmd.Parameters.Add(paramImageSize);

            SqlParameter paramImageData = new SqlParameter()
            {
                ParameterName = "@ImageData",
                Value = bytes
            };

            cmd.Parameters.Add(paramImageData);

            SqlParameter paramNewImageId = new SqlParameter()
            {
                ParameterName = "@NewImageId",
                Value = -1,
                Direction = ParameterDirection.Output
            };

            cmd.Parameters.Add(paramNewImageId);
```

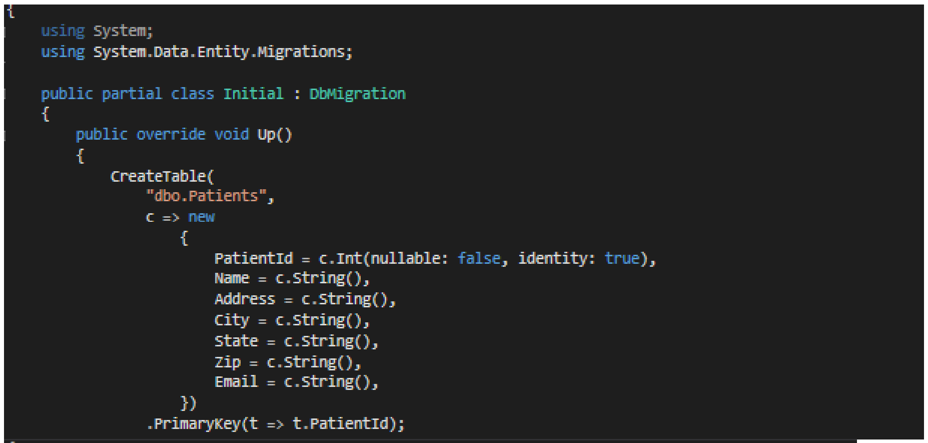


```
CREATE proc spUploadImage
@UserId nvarchar (128),
@ImageName nvarchar (255),
@ImageSize int,
@ImageData varbinary(max),
@NewImageId int output

as
Begin
    Insert into tblImages
    values (@ImageName, @ImageSize, @ImageData, @UserId)

    Select @NewImageId = SCOPE_IDENTITY()

End
```

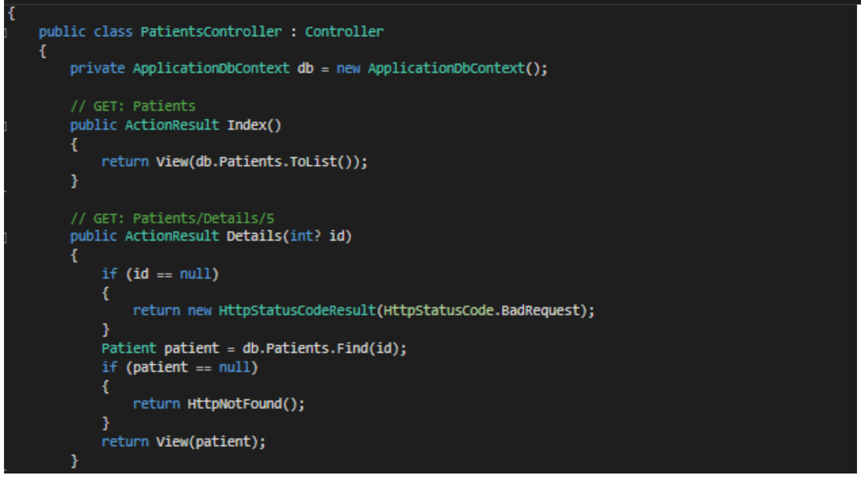
Once all of the above verification and authentication steps have been passed the user is then granted access to the database hosted on the web application containing the sensitive patient data as mentioned at the beginning of the implementation section only certain users have the privilege to edit and delete data. The database is created using models, entity first code migrations and controllers as you can see from the below code:



```
using System.ComponentModel.DataAnnotations;
using System.Globalization;
namespace Fyp.Models
{
    public class Patient
    {
        public int PatientId { get; set; }

        [StringLength(30, MinimumLength = 3)]
        public string Name { get; set; }

        [StringLength(30, MinimumLength = 3)]
        public string Address { get; set; }

        [StringLength(30, MinimumLength = 3)]
        public string City { get; set; }

        [StringLength(30, MinimumLength = 3)]
        public string State { get; set; }

        [StringLength(8, MinimumLength = 3)]
        public string Zip { get; set; }

        [StringLength(30, MinimumLength = 3)]
        [DataType(DataType.EmailAddress)]
        public string Email { get; set; }
    }
}
```



```
{
    using System;
    using System.Data.Entity.Migrations;

    public partial class Initial : DbMigration
    {
        public override void Up()
        {
            CreateTable(
                "dbo.Patients",
                c => new
                    {
                        PatientId = c.Int(nullable: false, identity: true),
                        Name = c.String(),
                        Address = c.String(),
                        City = c.String(),
                        State = c.String(),
                        Zip = c.String(),
                        Email = c.String(),
                    })
                .PrimaryKey(t => t.PatientId);
```

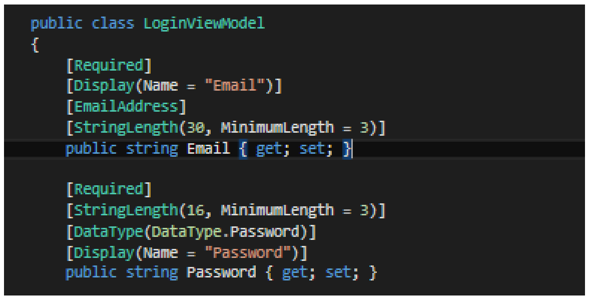


```
{
    public class PatientsController : Controller
    {
        private ApplicationDbContext db = new ApplicationDbContext();

        // GET: Patients
        public ActionResult Index()
        {
            return View(db.Patients.ToList());
        }

        // GET: Patients/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Patient patient = db.Patients.Find(id);
            if (patient == null)
            {
                return HttpNotFound();
            }
            return View(patient);
        }
```

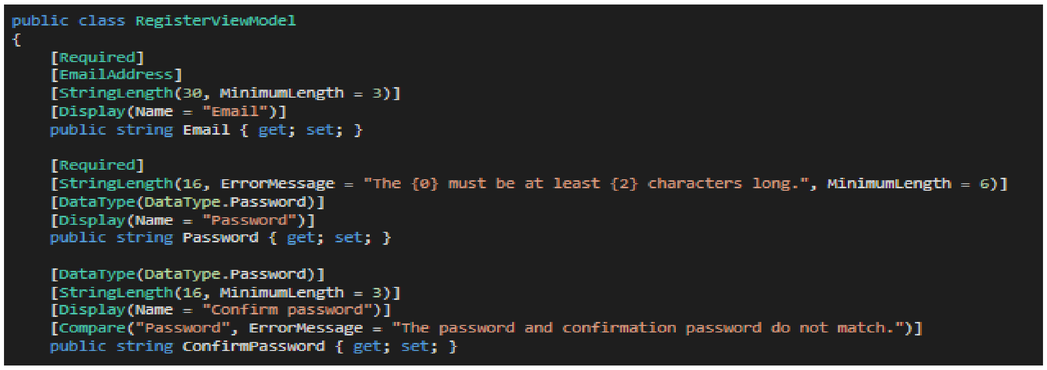
5.2 OWASP Secure Coding Principals

Data and Input Validation: The best way to protect against cross site scripting, command injection and buffer overflows is to constrain input by deciding what is allowed in the field and validating data – type, length, format, and range I have made use of these techniques in the below code on my login and registration view models input as this is the only data the user must input to register their account aside selecting their "Trusted Image". All data validation on a trusted system e.g. the application server. All validation failures result in input rejection. Validation is set up for expected data types, range and length.

- Prevents user from submitting excess amounts of data causing the application to exhibit unexpected behavior and prevents SQL injection.
- Risk: Database corruption, Unexpected behavior, Malicious queries, Effect integrity and availability.
- Mitigation Technique: Input Validation



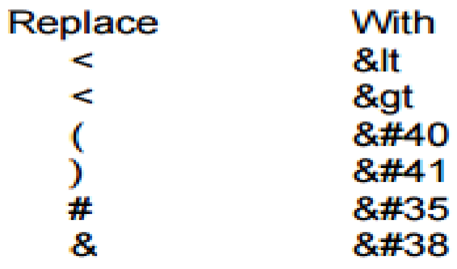
```
public class LoginViewModel
{
    [Required]
    [Display(Name = "Email")]
    [EmailAddress]
    [StringLength(30, MinimumLength = 3)]
    public string Email { get; set; }

    [Required]
    [StringLength(16, MinimumLength = 3)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }
```



```
public class RegisterViewModel
{
    [Required]
    [EmailAddress]
    [StringLength(30, MinimumLength = 3)]
    [Display(Name = "Email")]
    public string Email { get; set; }

    [Required]
    [StringLength(16, 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)]
    [StringLength(16, MinimumLength = 3)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    public string ConfirmPassword { get; set; }
```

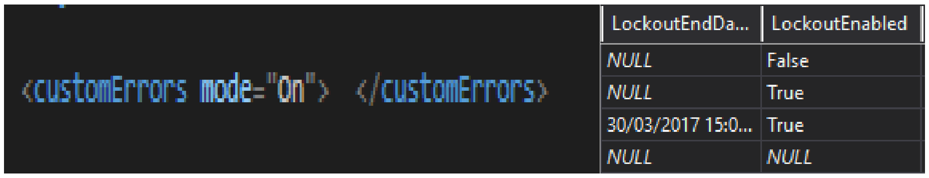
All my code will also need to be reviewed to protect against Cross site scripting this can be done by changing input variables that result in output and have no validation included. All headers, cookies, query strings, form fields, and hidden fields accepting input will need to be validated against acceptable data lists. Every input field must have a list of acceptable values.

Replacing the follow characters can also help to defeat XSS I will be applying this across my project also:



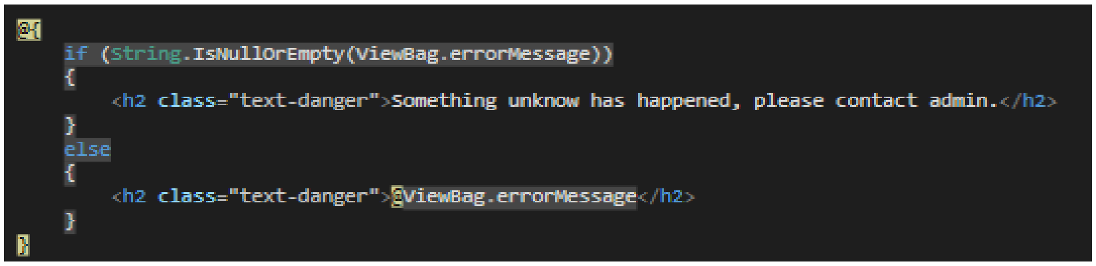
Replace	With
<	&lt
<	&gt
(	&#40
)	&#41
#	&#35
&	&#38

Output Encoding: Restrict disclosure of sensitive information in error responses, including system details, session identifiers or account information. Attacker can gain intimate knowledge of the inner workings of a system by studying error codes. To prevent this, we can generate custom errors to the user. All output encoding is done through a trusted system e.g. the application server.



```
<customErrors mode="On">  </customErrors>
```

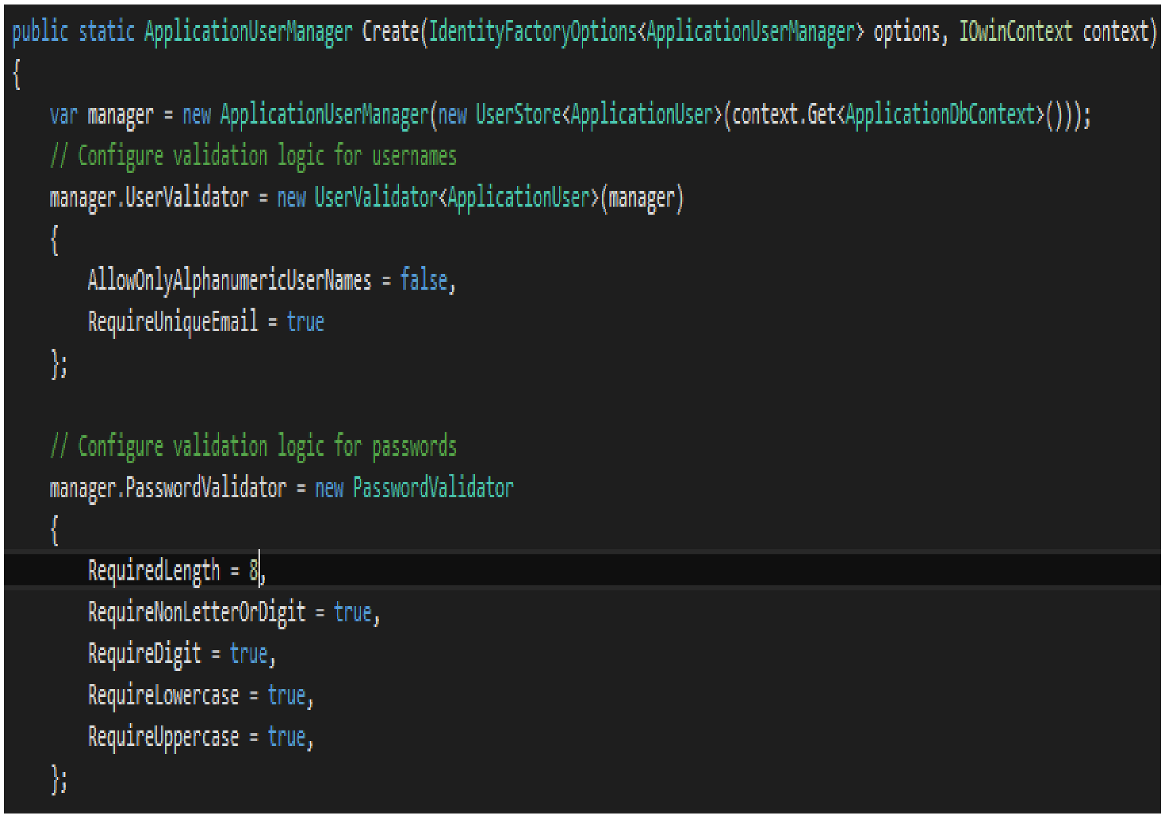
LockoutEndDa...	LockoutEnabled
NULL	False
NULL	True
30/03/2017 15:0...	True
NULL	NULL



```
@{
    if (String.IsNullOrEmpty(ViewBag.errorMessage))
    {
        <h2 class="text-danger">Something unknow has happened, please contact admin.</h2>
    }
    else
    {
        <h2 class="text-danger">@ViewBag.errorMessage</h2>
    }
}
```

- Restrict disclosure of sensitive information in error responses, including system details, session identifiers or account information.
- Risk: Attacker can gain intimate knowledge of the inner workings of a system by studying error codes.
- Mitigation Technique: Generate Custom Errors, Log Lockout Events.

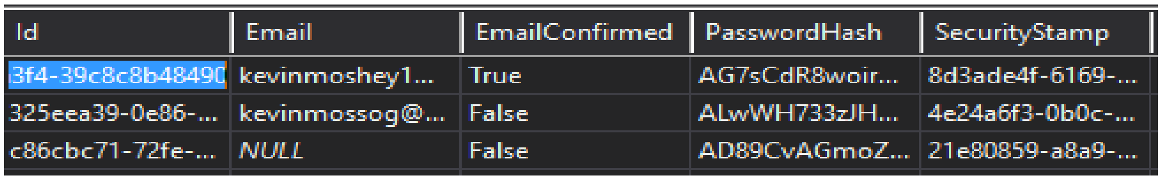
Authentication and password management: Authentication is the first line of defence and really is what my system is all about, I implemented the text based password and user name component, the parameters for this is defined below: I have set my system to only allow Alphanumeric user names and to require a unique e-mail to avoid duplication. The password length must be at least 8 characters, the password must contain a special character, a lower case and uppercase letter and at least one number as a minimum to generate a complex password that would take a considerable amount of time to brute force using dictionary/brute force attacks.



```
public static ApplicationUserManager Create(IdentityFactoryOptions<ApplicationUserManager> options, IOwinContext context)
{
    var manager = new ApplicationUserManager(new UserStore<ApplicationUser>(context.Get<ApplicationDbContext>()));
    // Configure validation logic for usernames
    manager.UserValidator = new UserValidator<ApplicationUser>(manager)
    {
        AllowOnlyAlphanumericUserNames = false,
        RequireUniqueEmail = true
    };

    // Configure validation logic for passwords
    manager.PasswordValidator = new PasswordValidator
    {
        RequiredLength = 8,
        RequireNonLetterOrDigit = true,
        RequireDigit = true,
        RequireLowercase = true,
        RequireUppercase = true,
    };
```

If the user wishes to change their password, they must enter their previous password first before being able to enter a new one. Once the user has successfully changed their password they will be sent back to the login screen and will need to log back in before being able to proceed.

All passwords and user ID's are encrypted using the SSL during transmission between the web app and the browser. All passwords and user ID's are hashed once they have been stored.

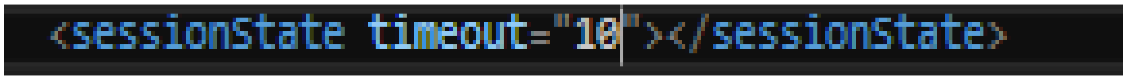


Id	Email	EmailConfirmed	PasswordHash	SecurityStamp
3f4-39c8c8b48490	kevinmoshey1...	True	AG7sCdR8woir...	8d3ade4f-6169-...
325eea39-0e86-...	kevinmossog@...	False	ALwWH733zJH...	4e24a6f3-0b0c-...
c86cbc71-72fe-...	*NULL*	False	AD89CvAGmoZ...	21e80859-a8a9-...

All authentication controls are enforced on a trusted system e.g. the application server. Authentication logic is kept separate from the resource being requested and redirection is used to and from the centralized authentication control. All authentication controls fail securely. As my application manages a credential store, it ensures that only cryptographically strong hashes of passwords are stored and that the table/file that stores the passwords and keys is write-able only by the application. Password hashing is implemented on a trusted system. Authentication data is submitted only once it has passed all data input validation. Only HTTP POST requests are used to transmit authentication credentials. Account disabling is enabled after an established number of invalid login attempts the account must is disabled for a period of time sufficient to discourage brute force guessing of credentials, but not so long as to allow for a denial-of-service attack to be performed.

- Prevents brute force and dictionary attacks, prevents user lockout, ensures users have valid details.
- Risk: User lockout, User fraud.
- Mitigation Technique: Complex Password, two factor authentication, E-mail confirmation, Password reset.

Session Management: The user will be provided with a session id, this id will be provided to the user and never selected by them, each session id will have a time limit attached as shown below, if the system is left idle and the token expires the user will be logged out and will need to re-authentication to continue their work, this will shorten the time window and attacker would have to potentially hijack a session. The session tokens are protected by the SSL. Session identities are created on a secure system. Logout functionality is available from all pages protected by authorization. A new session identifier is generated with every successful login.

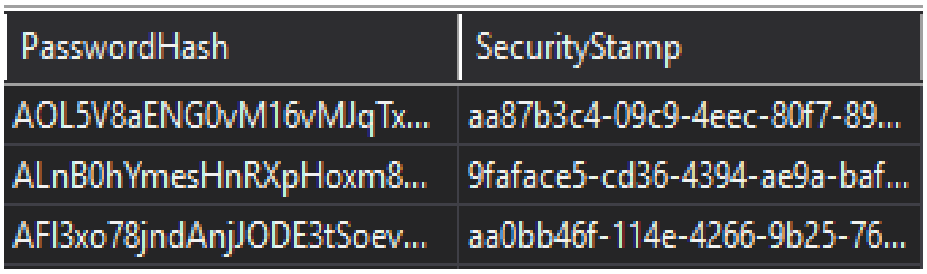

```
<sessionState timeout="10"></sessionState>
```

Authorization and access control: Access control is implemented on a trusted system in the form of an ACL with roles assigned to separate privileged and non-privileged users. Access controls fail securely being careful to keep source code hidden from malicious users. Access is restricted to protected functions, only authorised users can perform update and delete requests of data from the database.

Use of Cryptography:

My application currently used the form of default encryption implemented by MVC as a default to store user id’s and passwords as I had previously mentioned, however this encryption is using MD5/SHA1 two forms of hashing that are possible to crack both are known as broken algorithms as it is possible to crack both MD5 can be attacked and cracked in minutes using a common PC or laptop , while SHA1 would still need an extreme amount of computing power to be cracked 2^61 SHA1 calls this would need to be done by a large company or government agency however in the interest of security as the project continues I will be looking to change all encryption methods to AES 256 as it has yet to be cracked as is the current industry standard. All cryptographic functions are implemented on a trusted system. Master secrets are protected from unauthorised access. All hashes and encryption standards are intended to be un-guessable.

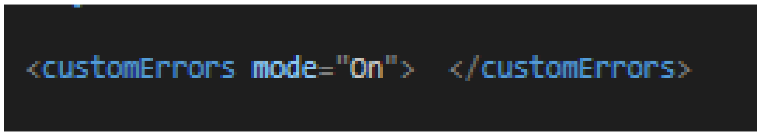
- Cryptographic functions used to protect secrets and sensitive data from the application users and attackers.
- Risk: Sensitive data lost / stolen, Data easy to read, Information easier to intercept and distinguish.
- Mitigation Technique: Encryption



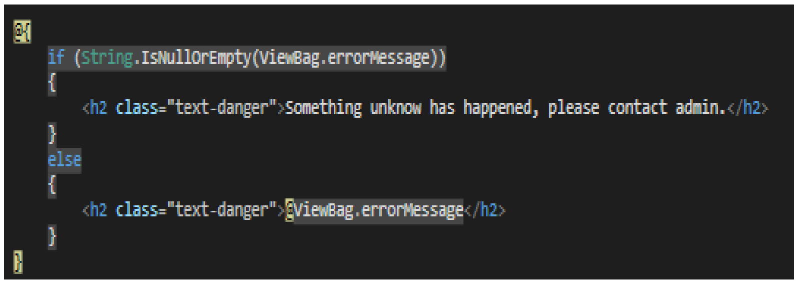
PasswordHash	SecurityStamp
AOL5V8aENG0vM16vMJqTx...	aa87b3c4-09c9-4eec-80f7-89...
ALnB0hYmesHnRXpHoxm8...	9faface5-cd36-4394-ae9a-baf...
AFl3xo78jndAnjJODE3tSoev...	aa0bb46f-114e-4266-9b25-76...

Error Handling and Logging: The system does not disclose information in error responses such as account information or session identifiers, generic errors have been implemented, login failure attempts are logged in the data base with a lockout functionality, once this has been tripped and an account is suspended the lock out count increased by 1 informed the DB admin that several unsuccessful login attempts have been made on a user-id.A cryptographic hash function is used to validate log entry integrity.

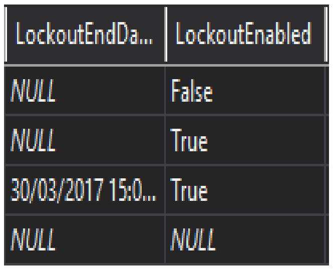
- Restrict disclosure of sensitive information in error responses, including system details, session identifiers or account information.
- Risk: Attacker can gain intimate knowledge of the inner workings of a system by studying error codes.
- Mitigation Technique: Generate Custom Errors, Log Lockout Events.



```
<customErrors mode="On">  </customErrors>
```



```
@{
    if (String.IsNullOrEmpty(ViewBag.errorMessage))
    {
        <h2 class="text-danger">Something unknow has happened, please contact admin.</h2>
    }
    else
    {
        <h2 class="text-danger">@ViewBag.errorMessage</h2>
    }
}
```

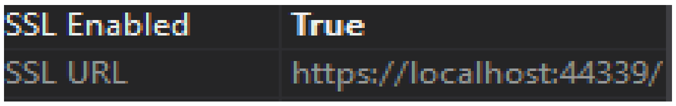


LockoutEndDa...	LockoutEnabled
NULL	False
NULL	True
30/03/2017 15:0...	True
NULL	NULL

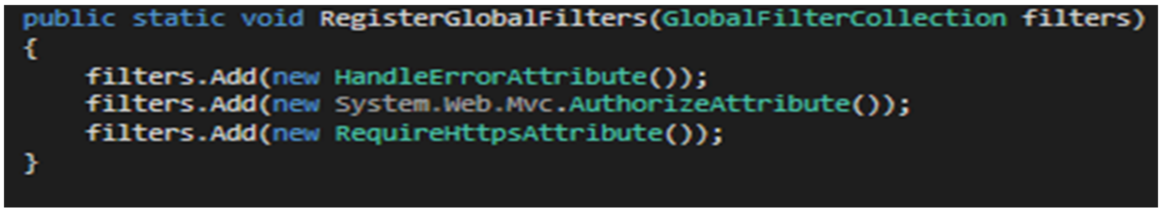
Data Protection: A standard of least privilege has been implemented restricting user functionality based on their assigned role. All highly sensitive data uses approved cryptographic practices to ensure the data cannot be intercepted and read. Server side code is protected through generic error generation. Sensitive information is not retrieved using HTTP GET requests. Auto Complete has been disabled when entering password fields and 2FA credentials. Access controls have been implemented to prevent modification of sensitive data.

Communication Security: SSL encryption has been enabled to protect the transmission of all sensitive information from man in the middle/sniffing attacks.

- Implementation of encryption for the transmission of all sensitive information.
- Risk: Sniff Traffic, Steal Credentials, Sensitive User Data, Man in the middle attack, Privacy, Session Token Interception.
- Mitigation Technique : Enabled SSL & Require HTTPs Attribute



SSL Enabled	True
SSL URL	https://localhost:44339/



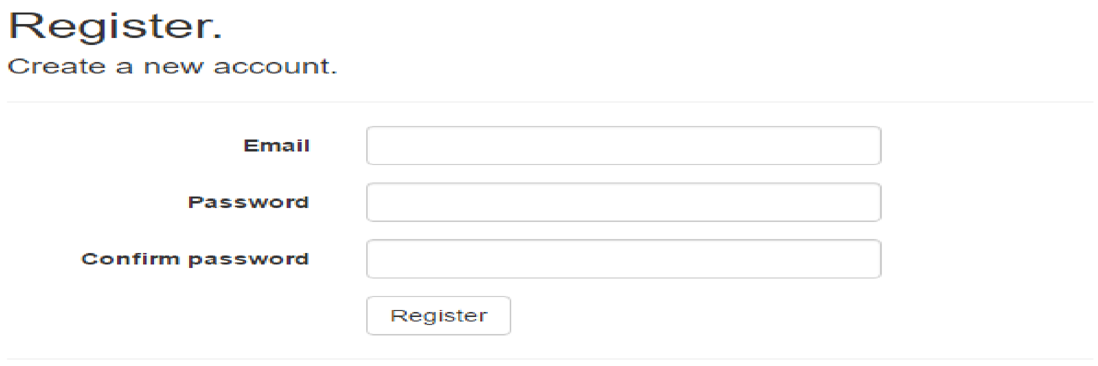
```
public static void RegisterGlobalFilters(GlobalFilterCollection filters)
{
    filters.Add(new HandleErrorAttribute());
    filters.Add(new System.Web.Mvc.AuthorizeAttribute());
    filters.Add(new RequireHttpsAttribute());
}
```

System Configuration: Servers, frameworks and system components are all up to date and have the latest fixes and patching applied. When exceptions of failures occur, they fail securely and provide as little information about the source/ project map as possible to the end user. Robot.txt has been disabled as to not disclose the directory structure. Any test code and functionality has been removed before final production of the application any unnecessary HTTP methods have been removed to simply the application and present a hardened and reduced attack surface to attackers.

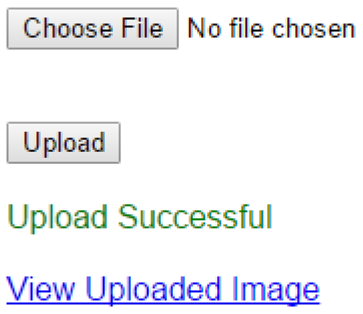
Database Security: Parameterised queries and stored procedures used throughout the database implementation of my project. Input validation and output encoding used to address meta characters**.** Secure credentials are needed to modify data held in the database. All default passwords have been removed, multifactor and complex password standards have been implemented on the system.

5.3 Graphical User Interface (GUI) Layout

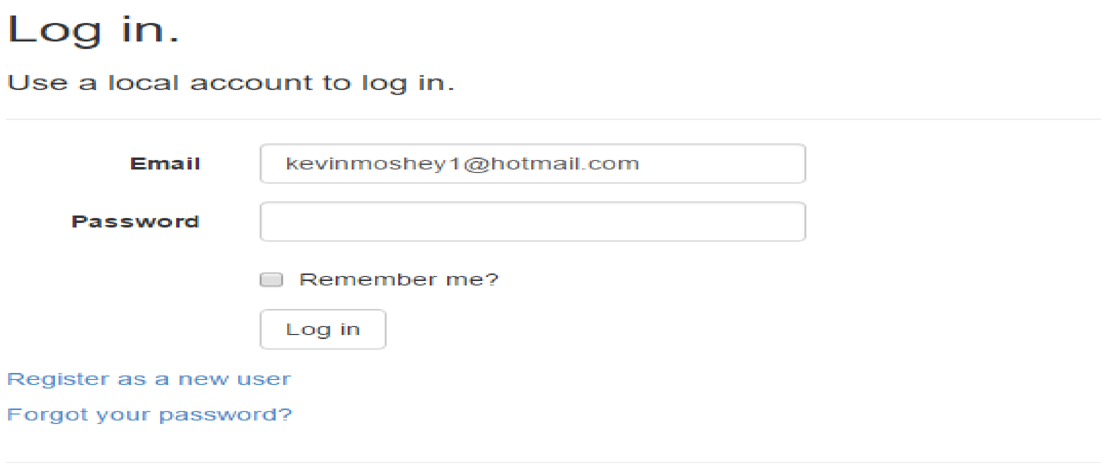
The below screen shows what the user will see and use if they have selected "Register as a new user", the e-mail field will check to make sure the e-mail is a valid format and the password field will make sure the password is at least 8 characters long, contains upper and lowercase, at least one number and a special character.



When a user click on choose image as "trusted image they are presented with this screen asking them to choose an image and then upload the image's metadata to be saved in the database.



The login screen of my application where the user can login once the e-mail account has been verified and they have passed the two-factor authentication, the user will then have access to the database. The user also has the option to register as a new user if they have not already done so, they can also use the forgot password link to have a password reset e-mailed to them.



Log in.

Use a local account to log in.

Email	kevinmoshey1@hotmail.com
Password	

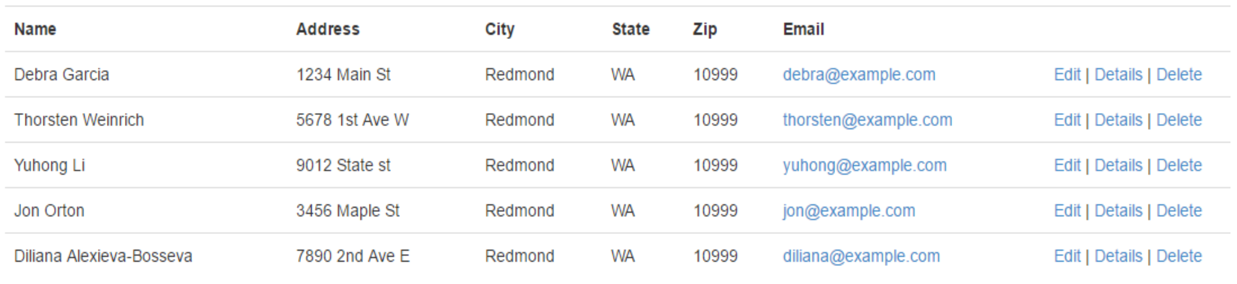
Remember me?

Log in

Register as a new user

Forgot your password?

The next screen is the database, currently it is only populated with dummy data, I will build a full database once the project has gone past the prototype stage, the database shows the “Sensitive data” my application will be protecting, it also offers “create, edit and delete” functionalities to the user, however only authorised accounts with a role assigned can use these functions. Any other accounts without the correct roles attempting to use these functions will be logged out and sent back to the login page.



Name	Address	City	State	Zip	Email	
Debra Garcia	1234 Main St	Redmond	WA	10999	debra@example.com	Edit | Details | Delete
Thorsten Weinrich	5678 1st Ave W	Redmond	WA	10999	thorsten@example.com	Edit | Details | Delete
Yuhong Li	9012 State st	Redmond	WA	10999	yuhong@example.com	Edit | Details | Delete
Jon Orton	3456 Maple St	Redmond	WA	10999	jon@example.com	Edit | Details | Delete
Diliana Alexieva-Bosseva	7890 2nd Ave E	Redmond	WA	10999	diliana@example.com	Edit | Details | Delete

5.4 Penetration Testing

I will be carrying out penetration or pen testing on my system once it has been completed, I can use this method to safely try to exploit vulnerabilities. These vulnerabilities may exist in operating systems, service and application flaws, improper configurations, or risky end-user behaviour.

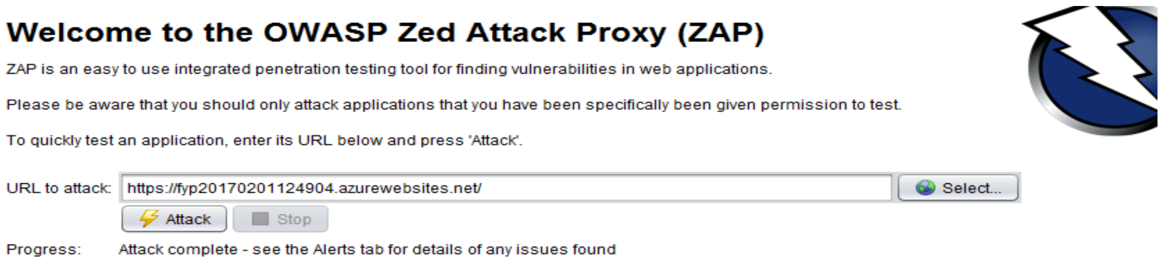
I will want to ensure that I have implemented strong input validation so that the listeners in my system will not terminate due to an invalid call length, I can test this requirement by implementing a “fuzz test” by calling the API with wrong number of parameters, with wrong data types, with values exceeding the allowed range, with special characters e.g. white spaces, control-sequences, characters. The listener shall be resistant against unexpected flood of requests (which could be a real unexpected load-peak, the result of errors in the calling application resulting in infinite number of retries, or an intentional attack).

I also want to protect against the listener terminating or stalling under an unexpected flood of requests. To counteract this I will enable the ability to count and limit the number of requests or open sessions accepted per source (server, IP-number), username if this is exceeded the system will return a defined error message. I can test my implementation of this counter measure by submitting more requests then the API can handle. The calling application will then need to treat this error message accordingly!

The database shall be protected from reaching the session limit caused by unexpected number of database connections, e.g. from application processes or interfaces opening a new database connection per request without closing / log out previous database session. If this is not mitigated against the database reaches the session limit and rejects new sessions resulting in a partial unavailability of service. Resulting in downtime for the user.

To carry out my penetration testing of my application I have decided to use OWASP Zap. The Zed Attack Proxy (ZAP) is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by people with a wide range of security experience and as such is ideal for developers and functional testers who are new to penetration testing as I am not an expert nor new to pen testing I would consider myself intermediate in penetration testing so OWASP ZAP meets my needs and experience level.

According to the scan from OWASP ZAP my application contains six low security risks and two medium security risks. I will do my best to find these errors and correct them as zero alerts would be preferable but very hard to achieve, given the limited time I have left to complete my project I will be looking to address the two medium security threats first and the rest if time allows for it.


Welcome to the OWASP Zed Attack Proxy (ZAP)

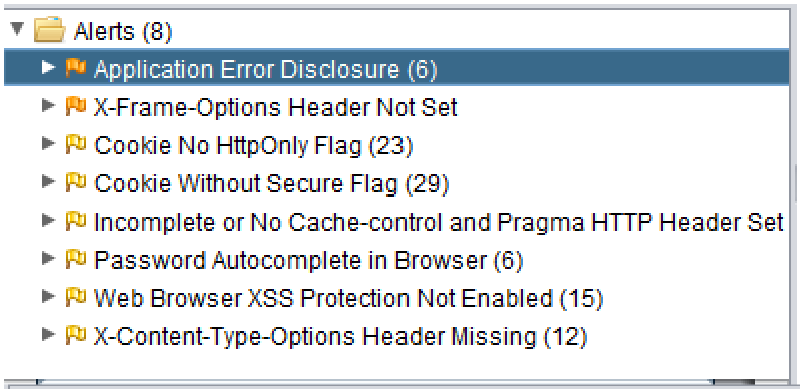
ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

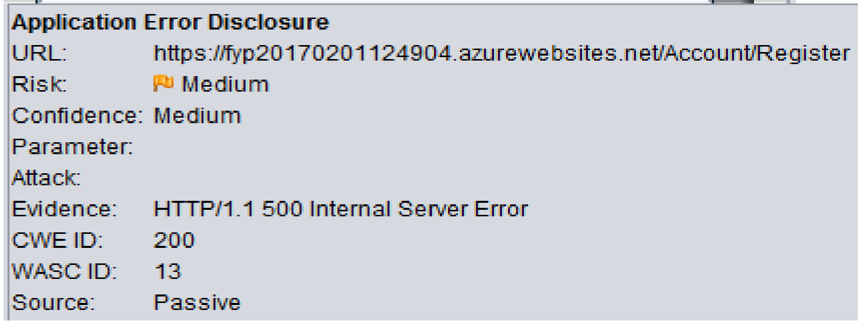
Please be aware that you should only attack applications that you have been specifically been given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: https://fyp20170201124904.azurewebsites.net/

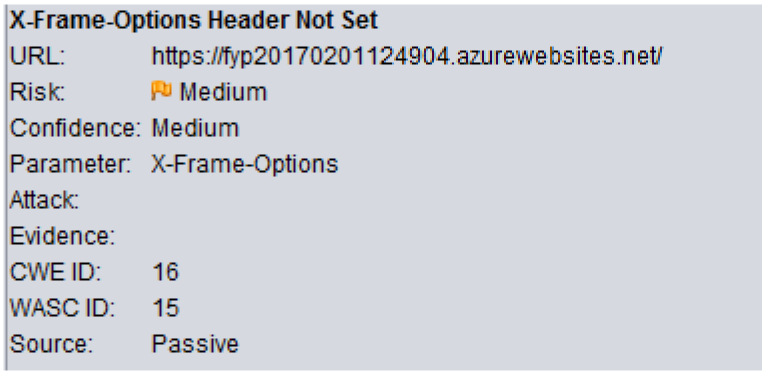
Progress: Attack complete - see the Alerts tab for details of any issues found


Alerts (8)
- Application Error Disclosure (6)
- X-Frame-Options Header Not Set
- Cookie No HttpOnly Flag (23)
- Cookie Without Secure Flag (29)
- Incomplete or No Cache-control and Pragma HTTP Header Set
- Password Autocomplete in Browser (6)
- Web Browser XSS Protection Not Enabled (15)
- X-Content-Type-Options Header Missing (12)


Application Error Disclosure

URL:	https://fyp20170201124904.azurewebsites.net/Account/Register
Risk:	Medium
Confidence:	Medium
Parameter:	
Attack:	
Evidence:	HTTP/1.1 500 Internal Server Error
CWE ID:	200
WASC ID:	13
Source:	Passive

This issue applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type.



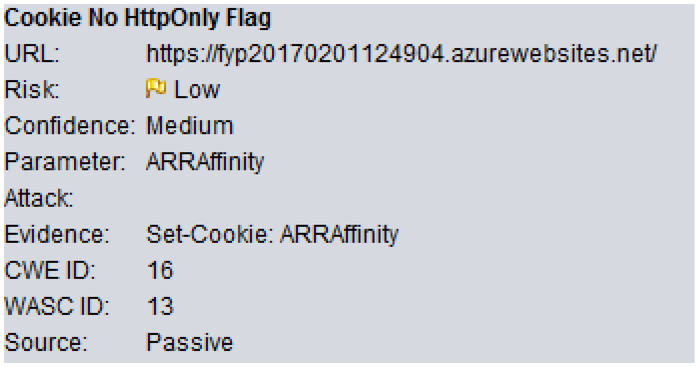
X-Frame-Options Header Not Set

URL:	https://fyp20170201124904.azurewebsites.net/
Risk:	Medium
Confidence:	Medium
Parameter:	X-Frame-Options
Attack:	
Evidence:	
CWE ID:	16
WASC ID:	15
Source:	Passive

Overall risk: Medium Description of error: The application reveals sensitive error which would allow a malicious attacker to gain internal working knowledge of the code and site map of the web application, this could be used to reverse engineer the code or to spot vulnerabilities in the code itself.

Likelihood: Medium Impact: The attacker is "hijacking" user interaction for the page they have in front of them and routing them to another page, most likely owned by another application, domain, or both. Can be used to steal information by placing opaque layers over buttons or text fields and having the information sent on to the attacker instead of the intended web page.

Recommendation: Ensure X-Frame-Options HTTP header is set on all web pages returned by the site.



Cookie No HttpOnly Flag

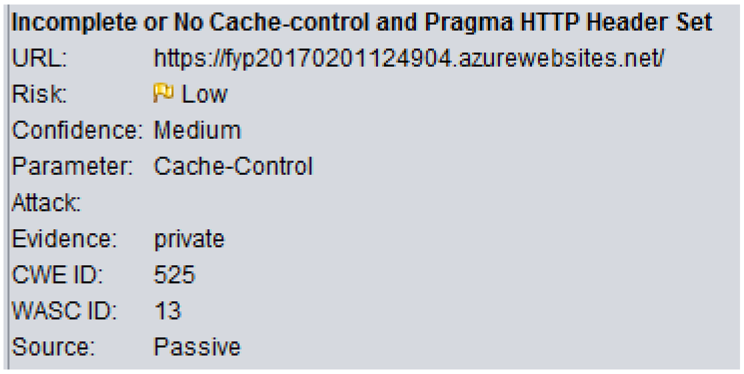
URL:	https://fyp20170201124904.azurewebsites.net/
Risk:	Low
Confidence:	Medium
Parameter:	ARRAffinity
Attack:	
Evidence:	Set-Cookie: ARRAffinity
CWE ID:	16
WASC ID:	13
Source:	Passive

Overall risk: Low

Description of error: Cookies can be accessed using JavaScript if a malicious script is run on the page the cookie can be accessed this can lead to the cookie being used on another browser leading to session hijacking.

Likelihood: Medium

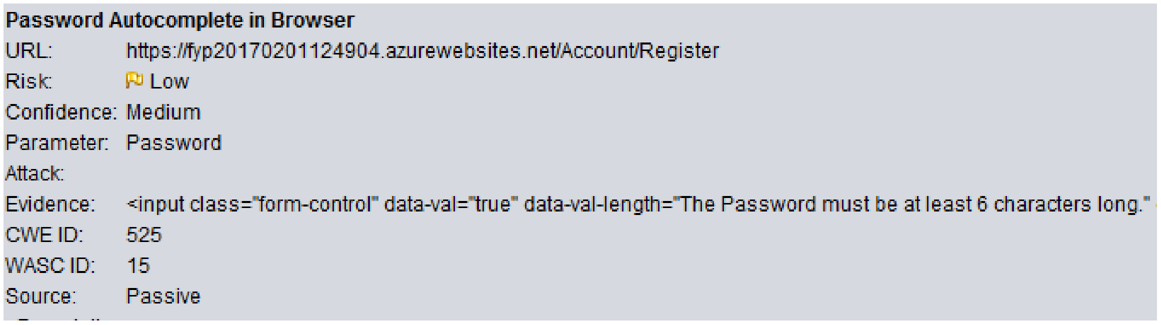
Impact: Session Hijacking - Impact of session hijacking is Severe, attacker can do anything that an Authentic user allowed to do on any website, depending on the session cookie that is stolen the user can gain Admin, Superuser or user rights. **Recommendation:** Ensure HTTP flag only is set for all cookies.



Incomplete or No Cache-control and Pragma HTTP Header Set

URL:	https://fyp20170201124904.azurewebsites.net/
Risk:	Low
Confidence:	Medium
Parameter:	Cache-Control
Attack:	
Evidence:	private
CWE ID:	525
WASC ID:	13
Source:	Passive

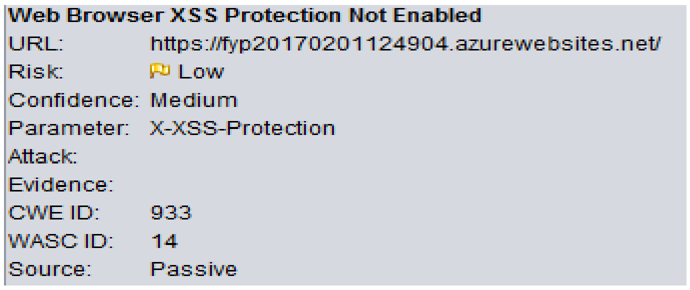
The cache-control and pragma HTTP header have not been set properly or are missing allowing the browser and proxies to cache content.



Password Autocomplete in Browser

URL:	https://fyp20170201124904.azurewebsites.net/Account/Register
Risk:	Low
Confidence:	Medium
Parameter:	Password
Attack:	
Evidence:	<input class="form-control" data-val="true" data-val-length="The Password must be at least 6 characters long."
CWE ID:	525
WASC ID:	15
Source:	Passive

The AUTOCOMPLETE attribute is not disabled on an HTML FORM/INPUT element containing password type input. Passwords may be stored in browsers and retrieved.



Web Browser XSS Protection Not Enabled

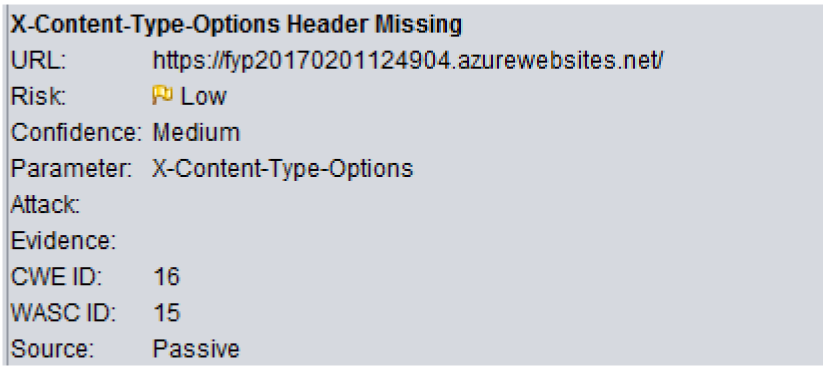
URL:	https://fyp20170201124904.azurewebsites.net/
Risk:	Low
Confidence:	Medium
Parameter:	X-XSS-Protection
Attack:	
Evidence:	
CWE ID:	933
WASC ID:	14
Source:	Passive

Overall risk: Low

Description of error: The web browser XSS prevention is not enabled by default or the X-XSS-Protect HTTP Response header has been disabled by the configuration of the server.

Likelihood: High

Impact: Disclosure of the user's session cookies, allowing the attacker to perform a session hijack and take over a user account, disclosure of end user files, trojan horse attacks or re-directs. Recommendation: Enabled the XSS Prevention filers by setting the X-XSS Protection HTTP response head to 1 on the web application



X-Content-Type-Options Header Missing

URL:	https://fyp20170201124904.azurewebsites.net/
Risk:	Low
Confidence:	Medium
Parameter:	X-Content-Type-Options
Attack:	
Evidence:	
CWE ID:	16
WASC ID:	15
Source:	Passive

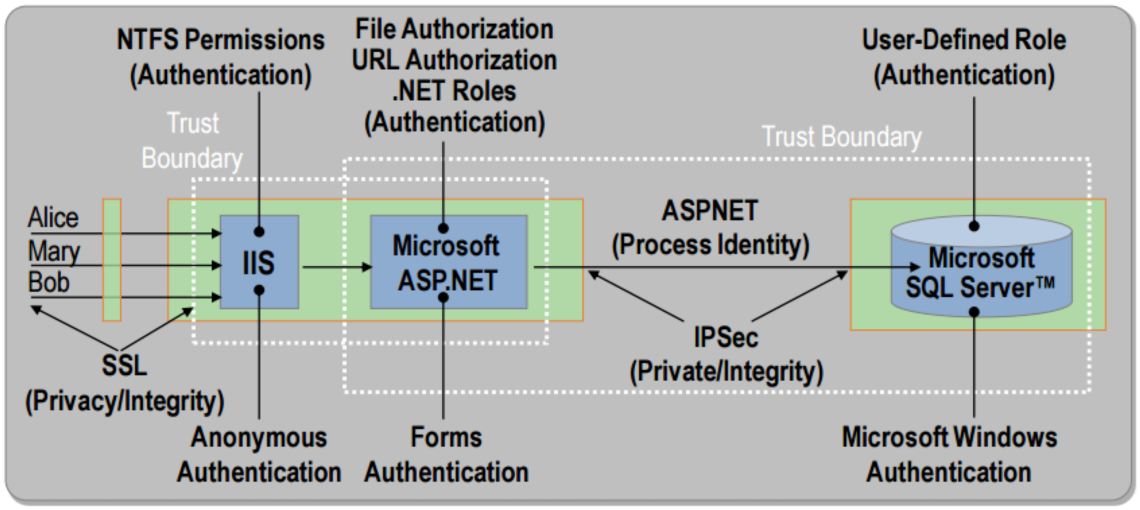
5.5 Evaluation

Evaluation

Threat Modelling

I will be using threat modelling during the planning phase of the project to optimize Network/ Application/ Internet Security by identifying objectives and vulnerabilities, and then defining countermeasures to prevent, or mitigate the effects of, threats to the project.

I will be following the SD3 threat model of Secure by Design focusing on secure architecture and code, threat analysis and vulnerability reduction. Threat analysis will allow me to evaluate the threats to my application aim to reduce overall security risks, find vulnerable assets, uncover vulnerabilities, identifies threats and give a solid base of security design specifications.


NTFS Permissions (Authentication)
File Authorization URL Authorization .NET Roles (Authentication)
User-Defined Role (Authentication)
Trust Boundary
Trust Boundary
Alice
Mary
Bob
IIS
Microsoft ASP.NET
ASPNET (Process Identity)
Microsoft SQL Server™
SSL (Privacy/Integrity)
IPSec (Private/Integrity)
Anonymous Authentication
Forms Authentication
Microsoft Windows Authentication

Average Application Response Time

This is the most important metric to understand how a website is performing from the user’s point of view. Thee application has been tested under different circumstances given the number of users working on the system simultaneously and the number of logins being requested at once.

Some factors that could not be tested, such as geographic location of the user, can affect the average response time for users and should be considered in an overall evaluation of application performance.

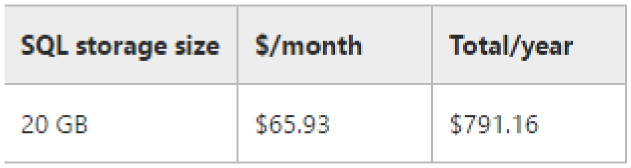
To ensure an optimal user experience, it’s essential to have an idea of what a round trip looks like for an application as my project is communicating between Visual studio 2015 and azure which work well together resulting in a low response time of 1-5 seconds.

Peak Response Time

The average response time gives you a sense of performance from the user’s perspective, the peak response time metric will help identify areas where performance could be improved. The peak response time for my system running on a local host is about 3 seconds, however when the project is deployed to azure the peak response time becomes about 30 seconds as the server my application is being hosted on is in North America causing a very high latency.

The peak response time metric pinpoints slow elements within the application that should be investigated and corrected by changing my host server.

My application stores data in a local SQL database, when run on azure the database uses azure to host the data, I estimate roughly each user record to take up 2 KB of space, based on an average of 65 percent of user logging in 10 times per month with the data being held for 7 years the storage requirement is about 16 GB given the way SQL data is held we would aim for about 20GB of space. The cost of this breaks down to be:



SQL storage size	$/month	Total/year
20 GB	$65.93	$791.16

6 Conclusions

The advantages of my project are that it is securely coded to reduce the possible attack surface attack the user interface is easy to navigate requiring little to no training, the application as maintains a consistent up time of 99% of course any routine maintenance or patching will have to be executed outside of peak hours to maintain the integrity of the application. Confidentiality of all data transmitted and stored is maintained through SSL transport layer encryption and Cryptographic techniques used in the database where sensitive/confidential data will be stored.

The difficultly in creating any innovative technologies lie in the fact that will be a chance of unknown flaws, faults or failures within my system. As flaws generally effect the Confidentiality, Integrity and Availability's of the application I am hoping that this is not the case, faults and failures may occur as there may be some errors in my coding, data, specifications or processes etc. These problems only occur due to me limited knowledge of coding and application deployment, given enough time and additional expertise the application could be developed to be one hundred percent securely locked down and deployed.

The most difficult part of the whole project was the learning process of developing a new technology using the C# coding language and visual studio 2015 a coding environment I am unfamiliar with leading to a lot of issues with frustration and a steep learning curve on my part. I was a very interesting learning experience on my part as it was constant struggle and demonstrated an ability to adapt and overcome challenges in a IT environment.

7 Further development or research

I would like to continue development and research on this application outside of college once the final presentation and showcase have been completed, my hope is that someone in the industry takes an interest in the new security mechanism I have developed and see the advantage that my system offers to anyone wanting to provide extra confidentiality, integrity and availability to their existing system or use my system as building block for developing future systems. Ideally, I would like to refine the application so that it can be a downloadable package that can be easily installed on any system business or personal.

I would also like to develop the application further so that it can be deployed on all know OS systems including MAC and Linux, I would also like to have my system implemented further into IOT devices as this is an emerging market in IT and has great potential for security to be applied to it.

I believe that if I can refine the system to the point it can be deployable by anyone with a level of IT knowledge, on any level of system be it local, distributed or mobile. I believe all data is precious whether it be photos of a loved one who has passed, past exam scores or essential and priceless business secrets, a person's data should remain confidential, integral and available, my hope is to develop a system that can ensure this.

8 Definitions, Acronyms, and Abbreviations

Access Control: A set of controls that grant or deny a user, or other entity, access to a system resource. This is usually based on hierarchical roles and individual privileges within a role, but also includes system to system interactions.

Attack Surface: The set of ways in which an adversary can enter a system and potentially cause damage.

Authentication: A set of controls that are used to verify the identity of a user, or other entity, interacting with the software.

Availability: A measure of a system's accessibility and usability.

Communication Security: A set of controls that help ensure the software handles the sending and receiving of information in a secure manner.

Confidentiality: To ensure that information is disclosed only to authorized parties.

Cross Site Request Forgery: An external website or application forces a client to make an unintended request to another application that the client has an active session with. Applications are vulnerable when they use known, or predictable, URLs and parameters; and when the browser automatically transmits all required session information with each request to the vulnerable application. (This is one of the only attacks specifically discussed in this document and is only included because the associated vulnerability is very common and poorly understood.)

Cryptographic Practices: A set of controls that ensure cryptographic operations within the application are handled securely.

Data Protection: A set of controls that help ensure the software handles the storing of information in a secure manner.

Database Security: A set of controls that ensure that software interacts with a database in a secure manner and that the database is configured securely.

Error Handling and Logging: A set of practices that ensure the application handles errors safely and conducts proper event logging.

Exploit: To take advantage of a vulnerability. Typically, this is an intentional action designed to compromise the software's security controls by leveraging a vulnerability.

Encryption: The transformation of data to hide its information content.

Failure: The inability of a system or component to perform its required functions within specified performance requirements.

General Coding Practices: A set of controls that cover coding practices that do not fit easily into other categories.

Hacker: Someone who violates computer security for malicious reasons, kudos or personal gain.

Impact: A measure of the negative effect to the business that results from the occurrence of an undesired event; what would be the result of a vulnerability being exploited.

Input Validation: A set of controls that verify the properties of all input data matches what is expected by the application including types, lengths, ranges, acceptable character sets and does not include known hazardous characters.

Integrity: The assurance that information is accurate, complete and valid, and has not been altered by an unauthorized action.

Mitigate: Steps taken to reduce the severity of a vulnerability. These can include removing a vulnerability, making a vulnerability more difficult to exploit, or reducing the negative impact of a successful exploitation.

Multi-Factor Authentication: An authentication process that requires the user to produce multiple distinct types of credentials. Typically, this is based on something they have (e.g., smartcard), something they know (e.g., a pin), or something they are (e.g., data from a biometric reader).

Output Encoding: A set of controls addressing the use of encoding to ensure data output by the application is safe.

Parameterized Queries (prepared statements): Keeps the query and data separate through the use of placeholders. The query structure is defined with place holders, the SQL statement is sent to the database and prepared, and then the prepared statement is combined with the parameter values.

Password: A secret series of characters used to authenticate a person's identity.

Sanitize Data: The process of making potentially harmful data safe through the use of data removal, replacement, encoding or escaping of the characters.

Security Controls: An action that mitigates a potential vulnerability and helps ensure that the software behaves only in the expected manner.

Security Requirements: A set of design and functional requirements that help ensure the software is built and deployed in a secure manner.

Session Management: A set of controls that help ensure web applications handle HTTP sessions in a secure manner.

System: A generic term covering the operating systems, web server, application frameworks and related infrastructure.

System Configuration: A set of controls that help ensure the infrastructure components supporting the software are deployed securely.

Threat: Something that could cause harm to a system or organization.

Trust Boundaries: Typically, a trust boundary constitutes the components of the system under your direct control. All connections and data from systems outside of your direct control, including all clients and systems managed by other parties, should be consider untrusted and be validated at the boundary, before allowing further system interaction.

Vulnerability: A weakness that makes the system susceptible to attack or damage.

Whitelist: A list of entities that are considered trustworthy and are granted access or privileges.

9 References

[1] Robust and secure image hashing - IEEE Xplore Document. 2016. *Robust and secure image hashing - IEEE Xplore Document*. [ONLINE] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1634363. [Accessed 07 December 2016].

How to Save Images in MySQL Database Using C#. 2016. *How to Save Images in MySQL Database Using C#*. [ONLINE] Available at: http://www.c-sharpcorner.com/uploadfile/deepak.sharma00/how-to-save-images-in-mysql-database-using-c-sharp/. [Accessed 07 December 2016].

AES encryption. 2016. *AES encryption*. [ONLINE]

Available at: http://aesencryption.net/. [Accessed 07 December 2016].

A MEDICAL RECORD DATABASE SYSTEM | Tosin Amuda - Academia.edu. 2016. *A MEDICAL RECORD DATABASE SYSTEM | Tosin Amuda - Academia.edu*. [ONLINE] Available at: http://www.academia.edu/3080257/A_MEDICAL_RECORD_DATABASE_SYSTEM. [Accessed 07 December 2016].

Rick-Anderson. 2016. *Create an ASP.NET MVC app with auth and SQL DB and deploy to Azure App Service*. [ONLINE] Available at: https://docs.microsoft.com/en-us/azure/app-service-web/web-sites-dotnet-deploy-aspnet-mvc-app-membership-oauth-sql-database. [Accessed 07 December 2016].

RickAndMSFT on Azure & MVC. 2016. *Securing your ASP.NET MVC 4 App and the new AllowAnonymous Attribute – RickAndMSFT on Azure & MVC*. [ONLINE] Available at: https://blogs.msdn.microsoft.com/rickandy/2012/03/23/securing-your-asp-net-mvc-4-app-and-the-new-allowanonymous-attribute/. [Accessed 07 December 2016].

Gail Zemanek Bayse. (2004). A Security Checklist for Web Application Design. Available: http://www.giac.org/paper/gsec/3725/security-checklist-web-application-design/106006. Last accessed 20th February 2017.

OWASP Secure Coding Practices - Quick Reference Guide - OWASP. 2016. *OWASP Secure Coding Practices - Quick Reference Guide - OWASP*. [ONLINE] Available at: https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide. [Accessed 11 December 2016].

Microsoft ASP.NET Site. 2016. *Create a secure ASP.NET MVC 5 web app with log in, email confirmation and password reset (C#) | The ASP.NET Site*. [ONLINE] Available at: https://www.asp.net/mvc/overview/security/create-an-aspnet-mvc-5-web-app-with-email-confirmation-and-password-reset. [Accessed 07 December 2016].

Microsoft ASP.NET Site. 2016. *Best practices for deploying passwords and other sensitive data to ASP.NET and Azure App Service | The ASP.NET Site*. [ONLINE] Available at: https://www.asp.net/identity/overview/features-api/best-practices-for-deploying-passwords-and-other-sensitive-data-to-aspnet-and-azure. [Accessed 07 December 2016]

Mike Brind. (27 January 2015). Working With Files. Available: http://www.mikesdotnetting.com/article/259/asp-net-mvc-5-with-ef-6-working-with-files. Last accessed 7th Jan 2017.

OWASP. (2010). OWASP Secure Coding Practices Quick Reference Guide. Available: https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf. Last accessed 17th Feb 2017.

Rick Anderson. (2013). Adding Validation. Available: https://www.asp.net/mvc/overview/getting-started/introduction/adding-validation. Last accessed 04/04/2017.

Saineshwar Bageri. (2016). 10 Points to Secure Your ASP.NET MVC Applications.. Available: https://www.codeproject.com/articles/1116318/points-to-secure-your-asp-net-mvc-applications. Last accessed 2nd Jan 2017.

Sharon Bell. (2015). 6 Critical Web Application Performance Metrics to Consider. Available:https://www.cdnetworks.com/blog/6-critical-web-application performance-metrics-to-consider/. Last accessed 02/05/2017.

Bharath. (2017). *Evaluating Cloud Hosting Costs.* Available: https://msdn.microsoft.com/en-us/library/ff803372.aspx. Last accessed 07/05/2017.

OWASP. (November 2010). OWASP The Open Application Security Project. Available:https://www.slideshare.net/LPetit/owasp-secure-coding-practices-quick-reference-guide. Last accessed 06/01/2017.

Mercury Consulting. (2012). Examples for Software Robustness Requirements. Available:http://www.it-checklists.com/Examples_Robustness_Requirements.html. Last accessed 03/04/2017.

10 Appendix

10.1 Monthly Journals

Reflective Journal One X13562797

Student name: Kevin Moss
Program (e.g., BSc in Computing): BSHC in Computing
Month: Sept

You don't have to follow the suggested format. These sub-headings and questions below may help you to get the most out of this journal, but you are free to modify as you see fit. Through this journal you demonstrate that you are engaged with the process and that you can identify what you need to do or change to progress and succeed in this project. Upload one journal every month. Expected word count 300 words (of you own words).

My Achievements

This month, I was able to … Come up with a project idea

My contributions to the projects included …Presenting the elements of my idea to a panel of lecturers to be given the go ahead to persue my chosen topic as my 4th year project.

My Reflection

I felt, it worked well to …. Try and come up with an idea.

However, I was not successful in … Presenting the idea to the panel of lecturers and my proposal was denied.

My Reflect on the Month as a whole : This first few weeks of Semester one have been more difficult than I had originally thought they would be, coming up with a completely original design for a software project is not coming easily to me. I have made my presentation of a mobile application I had come up with 2 weeks ago.
The application was going to calculate the recommend drink a person has on a night out using such measures as weight, gender, height, alcohol percentage of their chosen drink and then calculate a total for the user to have a guideline. As this was far too simple for a final year project I decided to add GPS location to find takeaway, clubs , bars , hospitals , caredocs etc., this would involve the GPS and a database functionality to store locations phone numbers etc.
I also wanted to include a support page for anyone having problems with depression, alcohol etc.
The app would have been secured with secure coding methods, encrypted communication and user authorization so as to add elements of my Cyber Security specialization to the overall project.
However after my proposal I was told my idea is too simple, I thought the idea of the

project was to show what you have learned over your 4 years .
My project was rejected and now I am at a loss as to what to design, I will be asking my security lecturer when I return to college on Monday to see if she might be able to assist with an idea or if she has a project she would like built.

Intended Changes

Next month, I will try to … Enquire with my Cyber Security lecturer about a project I can work on.

I realised that I need to … Get a project from my lecturer as my idea was rejected.

Supervisor Meetings

Date of Meeting: n/a
Items discussed: n/a
Action Items: n/a

Reflective Journal Two X13562797

Student name: Kevin Moss

Programme (e.g., BSc in Computing): BSHC Cyber Security

Month: October

My Achievements

This month, I was able to … Settle on a project Idea and begin my research for my chosen project, including my project proposal document.

My contributions to the projects included …

I began my research into the various areas that the project will have to address, namely getting the base application built before I focus on adding the security aspects to the project.

I looked into getting an image reader so that I can separate different parts of the images to be stored in a file format in a database. I also created my base database for the application so that I can build upon it from there

I also set up a git hub repository and a google drive account so that I can work with my assigned supervisor when building the project.

My Reflection

I felt, it worked well to …. Do some additional research outside of college hours especially during the reading week as I found it less daunting knowing I had little other outside pressure and found I was able to focus solely on the project a lot easier. I am hoping this will be the same over the xmas holidays.

I am finding having the project on top of all my other smaller projects is difficult as it pushes the project to the last thing on my mind when it needs to be the first however as I go along I am finding that the smaller projects I am being asked to complete throughout the year are all more or less helping me develop smaller parts of my larger project especially, API and web dev as it is teaching me a lot about server database relationships and also my secure applications and programming class is helping me integrated previously unknown technology and techniques with ease.

The only class I feel to be absorbing a lot of my time and giving little/nothing back is Artificial intelligence, I can understand if I was studying gaming it would be a necessary component to my 4th year course work but as I am not I cannot see its relevance, I am not the only one to feel this way among my peers. Hopefully the next round of 4th years will not have to deal with this as well.

Intended Changes

Next month, I will try to … Have the base application 100% running including the Web application itself and the first and second level of authentication.

I realised that I need to … Research further into how I am going to separate the various parts of an image into a database this is still very new and confusing to me as I have never worked in this area before.

Supervisor Meetings

Date of Meeting: 27/10/2016

Items discussed: Image reader/driver/converter

Several types of Blob storage for SQL

Breaking the image down into storable files

"Trusted image" As hash file

1st 2nd and 3rd level of authentication

Database to build / Web app to link it to.

Action Items: Code in the database.

Build the web application

1st and 2nd level of authentication

Research how to break image down

Reflective Journal Three X13562797

Student name: Kevin Moss

Programme (e.g., BSc in Computing): BSHC in Computing

Month: November

My Achievements

This month, I was able to … Conclude the initial research for my final year project, I was able to complete both my project proposal and my Requirements document for the software project, I also had more than 4 Ca's and other hand-ins to contend with so over all it has been a very busy month.

My contributions to the projects included …

I have managed to set out all of the function and non-functional requirements and have my project proposal submitted, I have begun with the base implementation of my project, so far, I have set up a database on the cloud, created a web application using MVC in Visual studio 2015 and built a secure registration and login page for my project. I have also been busy doing many other CA's, some of which have included programming for AI, a 7,000-word academic report and another project that thankfully is similar to my final year so that has been progressing along the same time frames.

My Reflection

I felt, it worked well to …. Set myself manageable goals and break the days of the week up as I work a part time job I lose about 3 days of college time automatically so I must utilise my time while in college as best I can, I devote 2 days to work on documents other theory heavy subject anything that requires an essay or a report etc. I find my strength lies in documentation so this is not a problem apart from the amount of time that the research for these documents can consume. I then devote whatever home time I have left to either coding or to studying for my Christmas exams, sadly more to coding than to studying but I plan on making up my usual notes once all my work has been submitted so I do not feel too far off where I want to be.

However the 10+ hour days I put in for the college only days and the 13+ hour day I am working are having an effect on my physical and mental health but it is only for the next few weeks and then a small break for Xmas which I am looking very forward to. The constant headaches and tiredness are beginning to take their toll.

However, I was not successful in …

The only part of this moth that I have not been successful as I would like in is AI once again, the difficulty and lack of step by step help I find very very frustrating especially as it is a very time consuming subject even if you do know what you are doing, I have submitted ¾ of the CA's sadly the final CA eludes me as we had to start again with a new code base which I find difficult at the best of time to understand. Web API also seems to be a project with a difficulty curve that never seems to cease, however I am just doing my best where I cannot maintain perfection. It is better to do something than nothing I suppose.

Intended Changes

Next month, I will try to … spend more time on my college work especially the final year project as I have the prototype presentation in 2 and a bit weeks therefore this will be taking priority, I will also need to have my technical report due within the next 9 days.

If I can balance the two of these I will be doing well, all the other CA's are due in the same week so I'll just have to see….

I realised that I need to … grow more arms and a larger brain oh and invent time travel.

Supervisor Meetings

Date of Meeting: 3/11/2016, 15/11/2016, 22/11/2016, 29,11/2016

Items discussed: Project Proposal Document, Project Requirements document, Prototype development, Technical Report Document.

Action Items:

What is needed for a project proposal.

Proof reading the proposal.

Project Requirement Document.

Project Requirement Document proof reading.

Proto type and mid-point discussion, what is needed what do we want to see, how to present.

Updates on prototype and midpoint presentation.

Reflective Journal Four X13562797

Student name: Kevin Moss

Programme (e.g., BSc in Computing): BSHC 4

Month: December

My Achievements

This month, I was able to … Put my prototype together and present it to my Midpoint panel of lecturers on being my academic supervisor and the other a random lecturer Joe Molumby to get an outside perspective on what he thought of my project.

My contributions to the projects included … Putting all of the prototype together, implementing secure coding practices, implementing a 2FA method using SMS , setting up a role based user set for my DB, Creating login, registration and database functionality, linking my database to the project. Password reset via e-mail etc. Also, to make a PowerPoint presentation about my prototype thus far and where it will be going in the future.

My Reflection

I felt, it worked well to provide a PowerPoint with all of the steps I have taken so far along with future implementation and to provide a work prototype and documentation to my panel of lecturers overseeing the midpoint.

However, I was not successful in … In fixing a small error I had where the role based user could not modify data as I had to drop and re-load my entire data base 2 days before the project presentation.

I do however feel that the marks I received for the quantity of work I presented were very low compared to some of my fellow students who go higher marks without having a working demonstration or and code to show their lecturer panel. As each student has different lecturers to view their project I suppose these discrepancies are going to happen as we are going off different sets of opinions for each person. I would apply for a formal review but It will not be checked until summer time at which point I don’t feel it will be relevant any longer, instead I will go over the presentation with my supervisor and find out where I lost marks.

Intended Changes

Next month, I will try to … finish my exams instead of writing reflective journals, this is time consuming while I have a further 3 subjects to study for, however the first exam went well so I am relatively hopeful the rest will be the same.

I realised that I need to … get back to studying and stop writing this journal.

Supervisor Meetings

Date of Meeting: 01/12/2016

Items discussed: Prototype Progress

Action Items: Have the Prototype ready for midpoint presentation or very close to. Good bit of a start into the document.

Reflective Journal Five X13562797

Student name: Kevin Moss

Programme (e.g., BSc in Computing): BSHC in Computing

Month: January

My Achievements

This month, I was able to … fix all of the known bugs and issues I had with my initial prototype. Set up versioning on GitHub and meet with my supervisor to discuss a deadline for implementation of the project.

My contributions to the projects included …

Fix all known bugs and issues and start this semester with a clean slate on the project to set up versioning so that I will always have this step to turn back to if something goes terribly wrong. Researching an image reader and making an image necessary for login as well as comparing the hash codes of the saved images.

My Reflection

I felt, it worked well to …. Start the project again from scratch as some of the errors were causing critical problems and were only made due to the trial and error process of learning a new language and working in an unfamiliar environment. As the project has gone on, I have impressed myself with how quickly I have picked up previously unknown technologies and practices.

However, I was not successful in … everything is more or less where I want it to be right now, I will finish implementation by the 15th of March hopefully and from there spend the rest of my time with the documents to get the best possible marks.

Intended Changes

Next month, I will try to … get the majority of the project finished and only have a few small things to tie off, leaving the SQL database to the end and then once that is linked, publish the application to azure so it is live in a cloud based domain and restore my Twilio for 2FA I just don't want to pay for it right now.

I realised that I need to … get the implementation out of the way so I can focus on other CA's and my new modules and the project documentation.

Supervisor Meetings

Date of Meeting: 02/02/2017

Items discussed: Progress on project from prototype, Midpoint presentation feedback.

Action Items: Set deadline for project implementation

Need a separate SQL database for medical records

Analysis and design document

Reflective Journal Six X13562797

Student name: Kevin Moss

Programme (e.g., BSc in Computing): BSHC 4

Month: February

My Achievements

This month, I was able to … begin working with some cloud storage resources, including how to work with blob storage, tables and general cloud storage connection methodologies.

My contributions to the projects included … Mapping the image metadata to my users table allowing them to hold the information relevant to their chosen trusted image, compare the tables between the original hash of the image and the one being uploaded.

My Reflection

I felt, it worked well to attend all of my supervisor meetings and to work on the project with whatever free time I have available even just having the project open while I watch TV allows me to keep looking at the code and seeing where I can make improvements or move some things around to ensure better performance and security for my application.

Intended Changes

Next month, I will try to … Have all of my main functionality finished and finish off the API side of the project I can then get back to writing my documentation with the remaining months' time.

I realised that I need to … carry out testing and add some further diagrams to my project once the implementation is completed, I am hoping to finish up by the end of March with the project implementation so that I can focus on my exams, presentation, showcase and documentation for the remaining time I have left in college.

Supervisor Meetings

Date of Meeting: 09/03/2017

Items discussed: Cloud technologies and Primary functionality of my application.

Action Items: Complete key functionality and begin designing the API.

Reflective Journal Seven X13562797

Student name: Kevin Moss

Programme (e.g., BSc in Computing): BSHC 4

Month: March

My Achievements

This month, I was able to … spend a lot of time trying to get the SQL database to join two tables from my ASPNETUSERS table to my Images table to assign a picture to each user, I have been trying to implement this key piece of functionality for over a month now and cannot seem to make any progress, so not really much in the way of achievements, I managed to get my buttons working and in the right place but that is about it for the last month.

My Reflection

I felt, it worked well to try every single piece of advice I could find on google, YouTube, and any advice I could get from my class mates and lecturers. However, I am still stuck where I was a month ago.

Intended Changes

Next month, I will try to … have my final year project ready for submission by the 8th of May and finish off my documentation.

Supervisor Meetings

Date of Meeting: 30/03/2017

Items discussed: SQL Database and table joins.

Action Items: Continue trying to get tables to join so we can compare and assign images.