National College of Ireland

BSc in Computing (Cyber Security)

Arnas Karciauskas

16457054

16457054@student.ncirl.ie

Software Project – DStore

Technical Report

https://github.com/ArnasKarciauskas/SoftwareProjectV2Solution

Table of Contents

Executive Summary

The objective of this project is to provide an application intended to be used by small organizations to store up to date information on employees and management of IT inventory. This allows easy and efficient management of company assets and employee details such as start dates, contact details etc.

Organization members can be assigned inventory devices to keep better tracking of where and when each device was assigned to.

The aim is to improve usability and efficiency of a business. Many organizations use old methods of inventory management and tracking such as paper copies or excel files which often get lost. The aim of this project is to remedy this issue and is focused on small business owners where there are budgetary concerns or no justification for expense of a commercial solution. It is difficult to convince smaller business owners to invest thousands into an advanced system in order to track small quantity of assets and staff.

Having worked with multiple IT companies, I realized that this is a real issue with data getting lost due to a corrupted hard drive or cleanup of offices which can result in permanently missing files.

The application hopes to address this problem by providing a secure online storage space, which can be accessed from anywhere with a network connection. Emphasis is placed on security by offering features such as 2 factor authentication, password requirements and input validation.

The application is developed using C# ASP.NET Core as it provides a high level of security for web applications.

1 Introduction

I was inspired to do this application after working in 3 different small-medium IT organizations. The organizations were well established but did not have an efficient way of tracking their asset inventory for engineers and customer data.

I thought of ways to improve this inefficiency and this applications hopes to address that. Many smaller organizations are unwilling to invest into expensive established solutions so this application provides a suitable alternative.

From real world experience these inefficiencies resulted in asset loss and financial loss to the company due to no information available anywhere.

I have also received feedback from the IT managers informing me that they would like such a solution to address these problems.

1.1 Background

Many organizations have poor tracking of their data which includes members of an organization and their assets. This can be written on paper which can often end up lost or stored on hard drive, which may become corrupted.

The reason for choosing this project is the potential to create this application which has an use in the real world scenario. I believe that many companies would find use in this as it would improve their data management and compliance. The application also would not require any kind of hardware. Administrators of the application would be able to access the application remotely and make changes to the application from anywhere.

This project will also help to improve my skills in cyber security by requiring me to implement secure principles in the application, which is a new area of information technology for me. It will also help me to refine my skills in web development.

1.2 Aims

Main goals of the project:

- Managing employee data
- Managing company assets
- Improve record keeping and ease of use
- Easy to use
- Emphasis on security
- Controlled by administrator using 2 factor authentication
- Ability for administrators to create, read, update and delete records

1.3 Technologies

C#: Is a programming language that is commonly used for web development. I have chosen this language as it is easier to implement and requires less code to achieve same objectives than other languages. I have greatly improved my skills by practicing and completing projects in 3rd year module “Advanced Internet Technologies” which have required the use of C# and Entity frameworks to complete a fully functional web application.

JavaScript: Is a programming language commonly used as part of web applications. The language will be used for implementing various scripts in the web application, calling different parts to action.

HTML: Hypertext Markup Language is the main language used for web sites. This language will be used for the front-end of the application.

Bootstrap: bootstrap is a popular HTML, CSS and JavaScript framework which was created to help develop responsive web pages focused on scalability and mobile.

CSS: Cascading Style Sheets is used for managing styling in HTML web pages. The language will be used for styling various aspects of the web application.

SQL: Structured Query Language is a language used for database management.

Visual Studio: is a development environment created by Microsoft. It is used for developing various software programs including web services and web applications, mobile apps etc. This development environment was chosen because of familiarity with it and optimization in making it easier to create web applications compared to other development environments. All of the application coding will be completed in Visual Studio IDE.

Excel: is a digital spreadsheet application part of the Microsoft Office suite. It will be used this for managing various stages of the project and creating deadlines. It will also be used for tracking testing results.

JavaScript libraries: Used for implementing QR code generation to enable 2 factor authentication.

1.4 Approach:

Will be using skills obtained in C# and performing further research on C# ASP.NET Core web framework. Using C# ASP.NET Core in Visual Studio to develop the application.
In addition, using skills obtained in “Security Principles” and “Secure Application Programming” modules and apply them during the application development to ensure that it meets the latest security requirements.
Implementation:
JavaScript libraries,
ASP.NET Core framework

1.5 Security approach:

As this application will store confidential information, it is important that common security vulnerabilities are addressed. OWASP Top 10 security vulnerabilities list will be used to address the top 5 security vulnerabilities. Security is a secondary thought for many organizations and too much of unnecessary security implementations can lead to organizations being discouraged from using it. The goal of this application is to make security efficient and make it seamless with the application.

Vulnerabilities addressed are:

1. Injection
2. Broken authentication
3. Broken access control
4. Cross-Site Scripting (XSS)
5. Sensitive data exposure/logging

1.5.1 Injection:

An injection is an attack type that modifies input which results in altered execution of the program. Injection is the most common vulnerability in any web application. OWASP TOP 10 shows that in survey done in both 2013 and 2017 the injection is #1 vulnerability in web applications. Injection works by an action performed by an attacker who inputs malicious code into an application, the application then takes the input as genuine and executes it. The script can provide the attacker with information such as full list of all the users with their login details, personal data, bank account details and other sensitive information.

Example of an injection would be an attacker running an SQL script that would GET the users from the application database. The main reason for injections being #1 in OWASP Top 10 2013 and 2017 is misconfigured or non-existing input validation allowing scripts to be injected into the database. As it is such a common and dangerous risk, it is of the highest importance that it is addressed by in the development of any web application.

Injection vulnerabilities can be addressed by proper implementation of the following security practices:

Input validation

Verification of user input. This can be in terms of length, format or type.

> **Emails:** the user should not be able to enter anything other than the correct format into the email field. The application throws an error if the format entered is not email such as “john@gmail.com”. If an user enters something like john.gmail.com, the application displays an error message to the user stating incorrect format.

Password rules: password should follow well established standards and use of weak passwords prevented. Samples of weak passwords are: admin/admin, root/password… The passwords should contain 8 characters, capital letters, numbers and special symbols. The system should not allow any unsecure passwords as they can be susceptible to password guessing attacks such as brute force attacks.

Principles of least privilege: in order to reduce the effect of a successful attack. The principle of least privilege should be applied. Users should only be given privileges explicitly and should not have any administration privileges by default. Example of this would be a regular user having the same level of privilege as administrator and the user being compromised would allow the hacker to perform all the actions. With proper implementation of principle of least privilege, if a hacker obtains non admin access, the impact would be reduced.

1.5.2 Broken authentication:

There are millions of valid username and password combinations that have been leaked and are publicly accessible obtained from previously compromised databases. Broken authentication means that the application is susceptible to a variety of authentication attacks. Malicious actors can gain access by impersonating an user or administrator and gaining full access to an organizational resources such as data and privileges that allow for changes such as retrieval of sensitive information or compromising the company resources in any other such as shutting down the systems, changing passwords etc.

Broken authentication can be addressed by proper implementation of the following security practices:

Multi factor authentication: 2 or 3 factor authentication can prevent the authentication of a malicious actor. If an attacker guesses the password, the attacker is prompted with an additional verification screen which is then sent to the genuine user to be confirmed. If the user is not the one attempting login, the user can refuse the connection preventing the malicious user gaining access.

Default credential use: Default credentials are widely available and are almost always used when attempting credential guessing attacks. Use of default credentials puts the organization at risk as usually the default credential user have admin privileges such as admin/admin. Malicious user logging in as administrator has full control of the application and can make any changes or retrieve any data. Applications should never use default credentials and always change the default username and password.

Weak passwords: Users that use weak passwords, are at a much greater risk of being guessed. Password rules should always be used to ensure maximum security and reduce the chance of successful attacks.

Lockout: lockout users after multiple failed attempts for a certain amount of time, this prevents brute force attacks.

1.5.3 Broken Access Control:

Unauthorized users gaining unauthorized privileges e.g. a regular user gaining administrator privileges and being able to modify parameters in the system. This is caused by lack of/improper configuration of security in an application.

Broken access control can be addressed by proper implementation of the following security practices:

Monitoring of malicious attempts: it is best practice to ensure that any failed attempts to access a resource are properly logged and can be accessed for later inspection. This can displays trends such as external IP attempting to access a system at multiple times a day at certain times. This allow threat response before the malicious attempt is successful.

Deny by default: deny all requests unless they are explicitly authorized. This is similar to principle how firewalls operate, traffic has to be explicitly authorized in order to pass through.

Directory listing: web application should be configured in a way that the metadata and backup files are no publicly accessible. They may contain information that potential attackers can exploit.

1.5.4 Cross-Site Scripting:

Cross-Site Scripting is a type of injection where malicious scripts are injected into genuine websites. Attacker uses the web application to send malicious code. The scripts comes from a genuine or “trusted” website therefore it is automatically executed. Cross-Site Scripting can provide a malicious actor access to information such as cookies and session data.

Cross-Site Scripting can be addressed by proper implementation of the following security practices:

Frameworks: frameworks that protect against cross-site scripting attacks should be used. There are many frameworks that provide protections, one example is ASP.NET Core framework which has explicit functions which protect against XSS.

Escaping: use of escape libraries that are proven to work against XSS.

Content Security Policy (CSP): Content security policy can reduce the effects of a successful XSS attack. CSP declares which resources are permitted to run depending on the source.

1.5.5 Sensitive Data Exposure/Logging:

Sensitive data exposure can be a result of lack of encryption of sensitive data or misconfiguration of the application which results in the display of sensitive information.

Poor logging is a security risk in that it affect the ability to identify ongoing or potential threats. Good logging can detect vulnerabilities before they appear.

Attackers are able to initiate the attacks by performing vulnerability probing. If an application is successfully probed, the vulnerabilities detected can be exploited. Security teams should perform their own tests to ensure that potential vulnerabilities are addressed.

Sensitive data exposure can be addressed by proper implementation of the following security practices:

Good logging configuration: it is best practice to gather as many logs as possible from all running systems so any possible malicious activity can be detected at any point. Logs can be investigated to find exactly what caused the issue. It is best practice to send logs to a remote location so if a system becomes compromised, the logs can still be accessed.

Monitoring: monitoring ensures that any unusual activities are monitored and breaches reported to the relevant party such as a security team so they can be addressed.

Testing: application should be thoroughly tested before deployment and after any changes are made to ensure that no new vulnerabilities appeared such as a change causing new vulnerabilities appearing. It is best practice to perform various tests in a test environment before deploying the application.

Other security implementations:

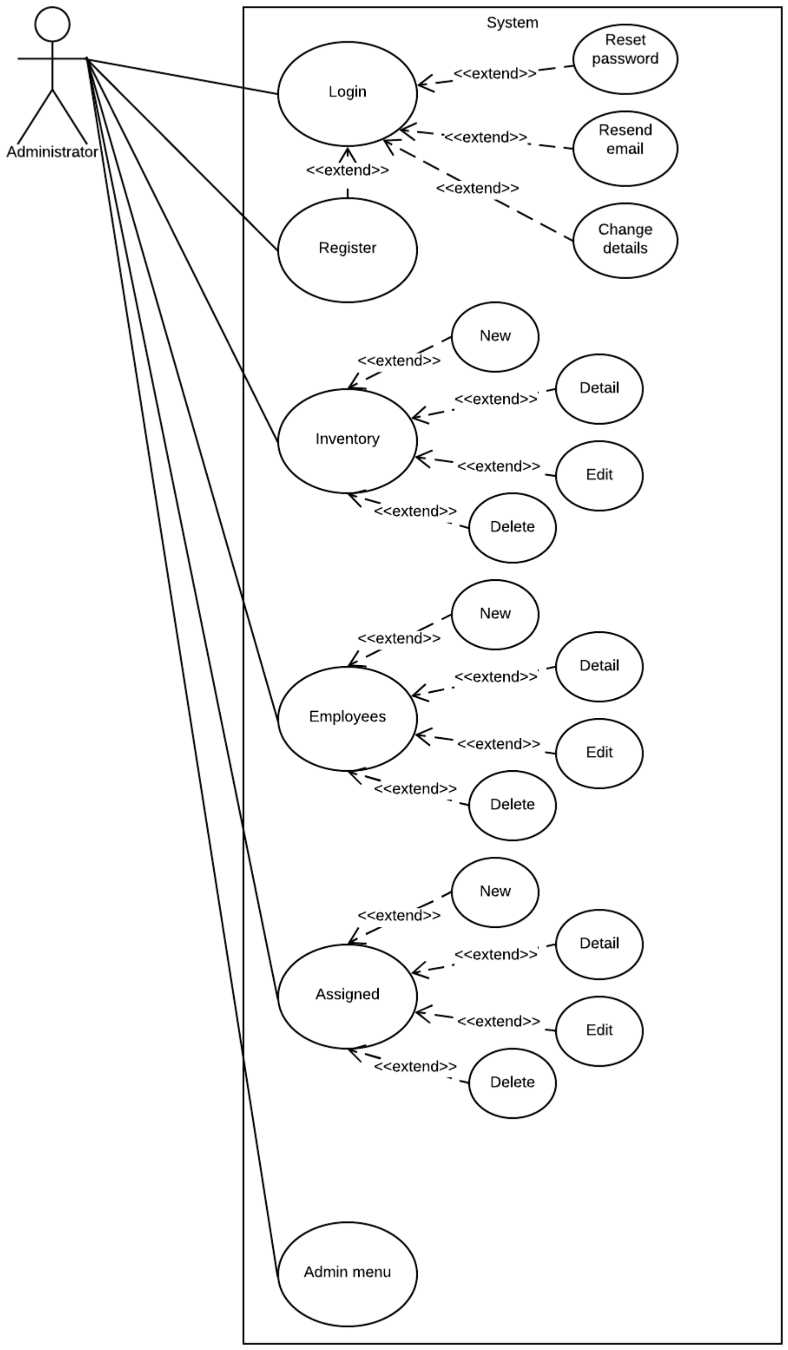
- Use of public and private key.
- Monitoring database activity.

Detailed security implementation is available in the section "2.4 Security Implementation".

2 System

2.1 Requirements

2.1.1 Use Case Diagram



2.1.2 Functional requirements

Requirement 1 <Registering Account>

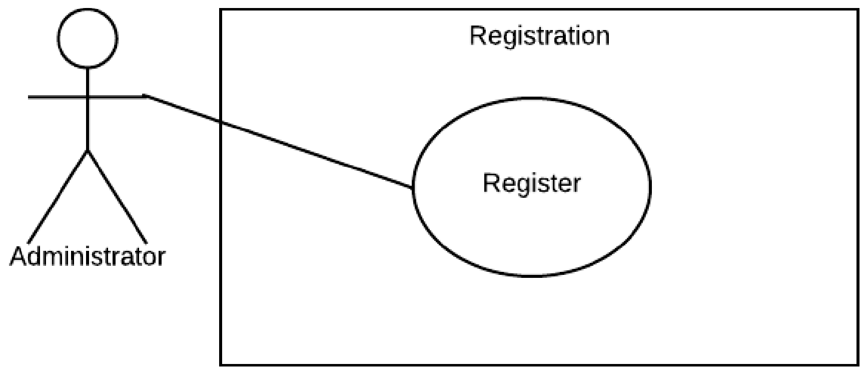
Description:

Users can register themselves in order to access the full features of the web application.

Scope:

The scope of this use case is to describe the process of registering an account to the web application.

Use case:



Flow Description: N/A

Precondition: N/A

Activation: Use case starts when the user clicks on the “Register” section.

Main flow:

1. Operator clicks on “Register” section.
2. Operator fills in email and password
3. Operator clicks “Register” button.
4. New user is created with full access privileges.

Alternate flow:

A1 <User already exists>

1. The system checks if user with an existing user email is present in the database.
2. System notifies the operator that the user already exists in the system.
3. Use case continues at position 2 of the main flow.

A2 <Incorrect information entered>

1. The system checks that correct information is entered in the fields.
2. The system notifies the administrator if the user details are entered incorrectly (empty, incorrect format (email)).
3. The use case continues at position 2 of the main flow.

Termination:

New user is created using the entered information, page restarts and operator is able to create another user.

Post condition:

Users can login with email and password.

Success Conditions:

1. User account gets created.
2. Created user can login to the web application.
3. User details are saved in the database.

Failure conditions:

1. User account is not created.
2. User cannot login to the web application.
3. User details are not saved in the database.

Requirement 2 <Login>

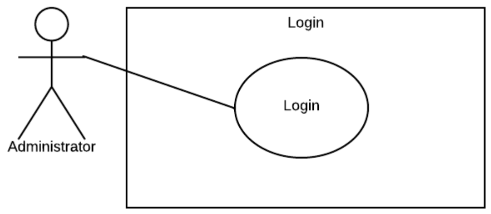
Description:

This requirement is essential to the web application as it only allows authenticated users to perform the functions of the web application, which are: viewing, creating, reading, updating and deleting.

Scope:

The scope of this use case is to show the interaction between the initiator and the application performing the login function.

Use case:



Flow Description: N/A

Precondition: Requirement 1 (Registration).

Activation: Use case starts when the user clicks "Login" section of the web application.

Main flow:

1. The user enters their email.
2. The user enters their password.
3. The user is logged in to the system.
4. The user is redirected to the home page.

Alternate flow:

A1 <Incorrect details entered>

1. The user enters incorrect email or password.
2. The system notifies the user that the email or password is incorrect.
3. Use case continues at position 1 of the main flow.

Termination:

T1 <Successful login>

1. Users enters their details correctly and is brought to the main page.

T2 <Locked out>

1. The user enters the credentials incorrect 3 times in a row.
2. The user is locked out for 5 minutes.
3. The user is unable to login.

Post condition:

Success Conditions:

1. Users gets logged in.
2. User can access full features of the application.

Failure conditions:

1. User cannot login to their account.
2. User cannot access user features.

Requirement 3 <Password reset>

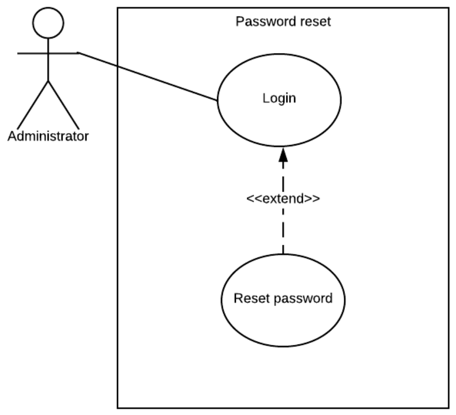
Description:

This requirement allows registered users to reset their password. The password reset link gets sent to their registered e-mail.

Scope:

The scope of this use case is to show the steps required to reset the password for a registered account.

Use case:



Flow Description: N/A

Precondition: Requirement 1 (Registration).

Activation: Use case starts when the user clicks “Login” section of the web application and “reset password” section.

Main flow:

1. The user enters their email.
2. System sends a password reset link to the entered users e-mail if it exists.
3. User clicks the link sent by the system
4. System prompts the user to enter a new password.
5. User enters a new password.
6. System changes the users’ password.
7. User gets redirected to the login page.

Alternate flow:

A1: <Non-existent employee>

1. The user enters a non-existent user email.
2. System notifies that the user will be sent an email if it exists (security reasons for no display if exists or not).
3. Use case continues at position 1 of the main flow.

A2: <Incorrect password format>

1. User clicks on the password reset link.
2. System prompts the user to enter a new password.
3. User enters password that does not meet the minimum requirements.
4. System prompts the required password characteristics.
5. Use case continues at position 4 of the main flow.

Termination:

Users successfully changes their password and is brought to the login page.

Post condition:

Success Conditions:

1. Users changes their password.
2. User can login to their account with the new password.

Failure conditions:

1. User is unable to change their password.
2. User cannot login to their account.

Requirement 4 <Changing user details>

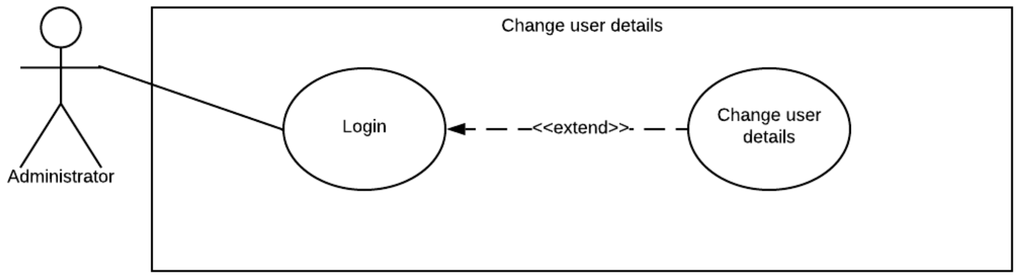
Description:

This requirement allows administrator to change all the user account details (email, password and phone number.).

Scope:

The scope of this use case is to show the steps required to change registered user details by administrator.

Use case:



Flow Description: N/A

Precondition: Requirement 1 (Registration), user must be logged in.

Activation: Use case starts when the user clicks “%User email%” display on the web application.

Main flow:

1. The user chooses a desired option to change details.
2. The user changes user details (email, phone number or password)
3. The user clicks “Save” button.
4. User gets redirected back to the Index page of the profile management section.

Alternate flow:

A1: <Incorrect changes>

1. The user enters incorrect details or incorrect format in one of the user fields (phone number incorrect format, email incorrect format or size…)
2. System notifies the administrator that the details entered are not correct or the format is wrong.
3. System displays the relevant error to the user.
4. Use case continues at position 4 of the main flow.

Termination:

User successfully changes user details and is redirected back to Index page of the profile management section.

Post condition:

Success Conditions:

1. User details are successfully changed.
2. User can login to their account with new details or have their other details updated.

Failure conditions:

1. User is unable to change user information.
2. User information stays the same.

Requirement 5 <Adding data to application>

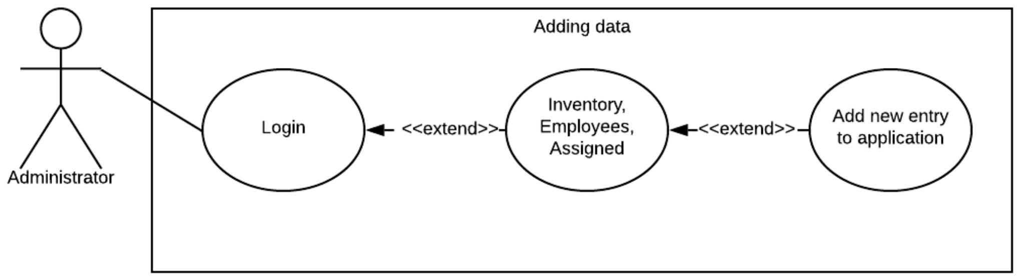
Description:

This requirement allows the user to add data to the web application.

Scope:

The scope of this use case is to show the steps required to add data by the user.

Use case:



Flow Description: N/A

Precondition: User must be logged in to the web application.

Activation: Use case starts when the user selects a section in which to add a new entry.

Main flow:

1. The user clicks on the desired "%sectionname%" button of the web application

2. The user selects “create new” button in the section.
3. The user fills out the required fields.
4. The user clicks “add new entry” button.
5. The entry is added to the database and is displayed on the index page.
6. User gets redirected back index page of the relevant section.

Alternate flow:

A1: <Incorrect format>

1. The user fills out the fields incorrectly due to formatting or size.
2. System displays the relevant error to the user.
3. Use case continues at position 3 of the main flow.

Termination:

User successfully adds a new entry to the web application and gets redirected back to the index page of the relevant section.

Post condition:

Success Conditions:

1. New entry is added to the web application.
2. Logged in users can view the new entries and make changes to the new entry.

Failure conditions:

3. User is unable to change add a new entry.
4. No new entries are added to the database.

Requirement 6 <Editing data entries>

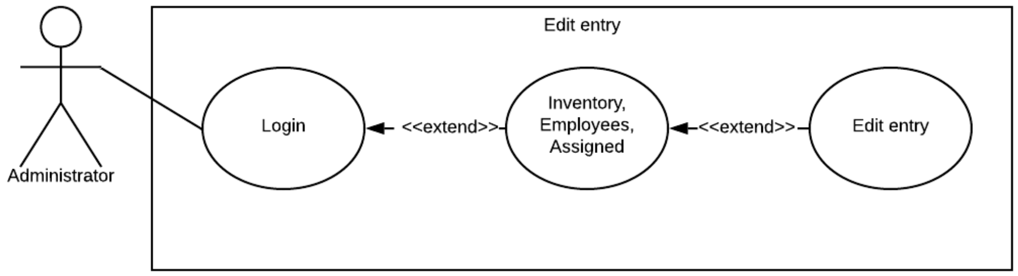
Description:

This requirement allows the user to edit existing entries that are already on the web application.

Scope:

The scope of this use case is to show the steps required to edit entries by the user.

Use case:



Flow Description: N/A

Precondition:

1. Entry must exist,
2. User must be logged in.

Activation: Use case starts when the user clicks "Edit" button on an existing database entry.

Main flow:

1. The user selects an entry to edit.
2. The user changes the details of the selected entry.
3. The entry gets updated in the application.
4. User gets redirected back to the index page of the selected section.

Termination:

User successfully edits an entry on the web application and is redirected back to index page of the selected section.

Post condition:

Success Conditions:

1. Edited entry is updated in the web application.
2. Logged in users can view the update entry.

Failure conditions:

1. User is unable to edit the entry.
2. Entry remains unchanged.

Requirement 7 <Deleting entries>

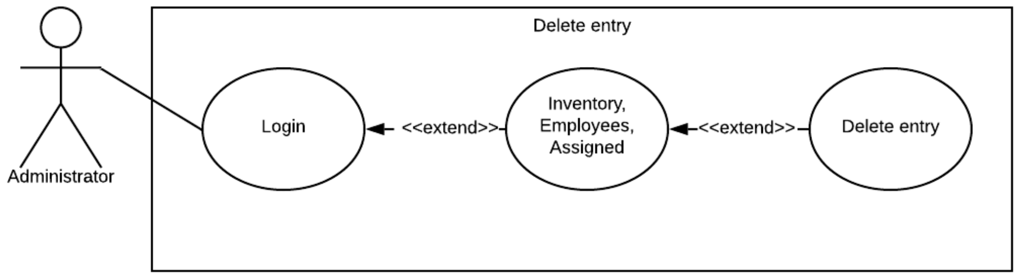
Description:

This requirement allows users to delete existing entries in the web application.

Scope:

The scope of this use case is to show the steps required to delete existing entries on the web application.

Use case:



Flow Description: N/A

Precondition:

1. Entry must exist,

2. User must be logged in.

Activation: Use case starts when the user “clicks” delete button on an existing entry.

Main flow:

1. The user selects an entry do delete
2. The system opens the document and displays it to the user.
3. User confirms the deletion by clicking the “confirm delete” button.
4. The system deletes the entry from the database.

Alternate flow: N/A

Termination:

User deletes the document.

Post condition:

Success Conditions:

1. User successfully deletes the document.
2. Database is updated to reflect the deletion.

Failure conditions:

1. User is unable to delete the document.
2. Database is not updated.

2.1.3 Data requirements

Data will be handled by using ASP.NET Core Entity Framework Object-Relational-Mapper. The reason for this choice is it saves a significant quantity of time because the data model is stored in one place and it is easier to update the code. Another benefit is that actions such as database handling are done automatically which reduces human error which can create security faults in the application which can be exploited by malicious actors.

2.1.4 User requirements

2.1.4.1 Devices:

Users will be able to access the web application through their mobile devices or any other device which has a browser (PC, tablets...).

Layout has been optimized for all devices using responsive web design principles.

2.1.4.2 Internet access:

For maximum security, the application should be hosted in an internal network. Users wanting to access the application are required to have an internet connection.

2.1.5 Usability requirements

To be able to access the full features of the web application, users will have to setup their own accounts by using the registration section of the web application.

Device must be connected to the internet throughout the process to use the functionality of the web application.

2.2 Non-Functional Requirements

2.2.1 Performance/Response time requirement

The performance of the application will mainly depend on the internet speed of the application user. Most of the data inputted will be text format therefore no extensive resources are demanded by the web application. Even low performance networks are able to handle the maximum load of the application.

There should be not issues with performance and this will be validated by using various application tests.

2.2.2 Availability requirement

The application will need to be always available, as it will store important information, which may need to be accessed at any time during events such as an external audit. Availability failure would defeat the purpose of the application which is to maintain a database of records accessible from anywhere.

2.2.3 Recovery requirement

The application users will have the possibility to reset their passwords and download their data.

The application should be regularly backed up by the owner in case of outages for whatever reason. Backups can be tape backups or hosting provider.

2.2.4 Security requirement

Any application user will be required to have a strong password (Capital letter, symbol, number and have a minimum length of 8 characters). Passwords are encrypted using hashing and salting.

Application is used for storing confidential data therefore best security practices need to be followed to provide adequate security features and maintain. OWASP top 10 is used for guidance as to which security features are the most important.

2.2.5 Reliability requirement

The reliability of the system is essential as important organizational information will be stored on the system. The system should not have any bugs and be stable. Testing will be carried out to find any possible bugs in the system.

2.2.6 Maintainability requirement

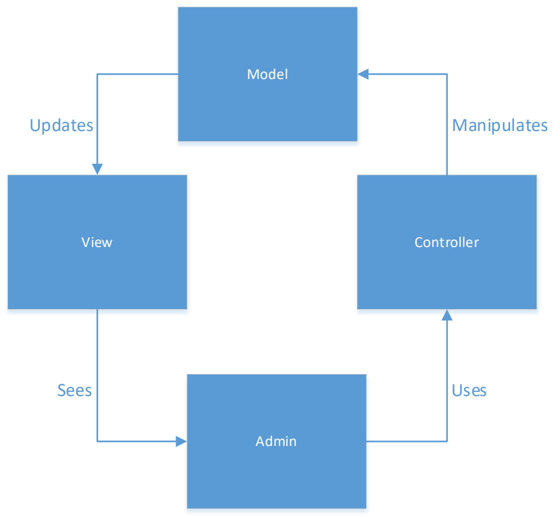
The advantage of using C# ASP.NET Core framework is code reusability. Good coding practices will be used to ensure that the application is easy to understand and maintain. Software code is commented to ensure easy understanding by any party who may want to inspect it or make changes such as extra functionality or bug fixing.

2.2.7 Reusability requirement

To comply with best programming practices, reuse of code will be implemented. Reuse of will reduce overall time required to complete objectives. This includes variables and security requirements.

2.3 Design and Architecture

ASP.NET Core model, view, controller principle:



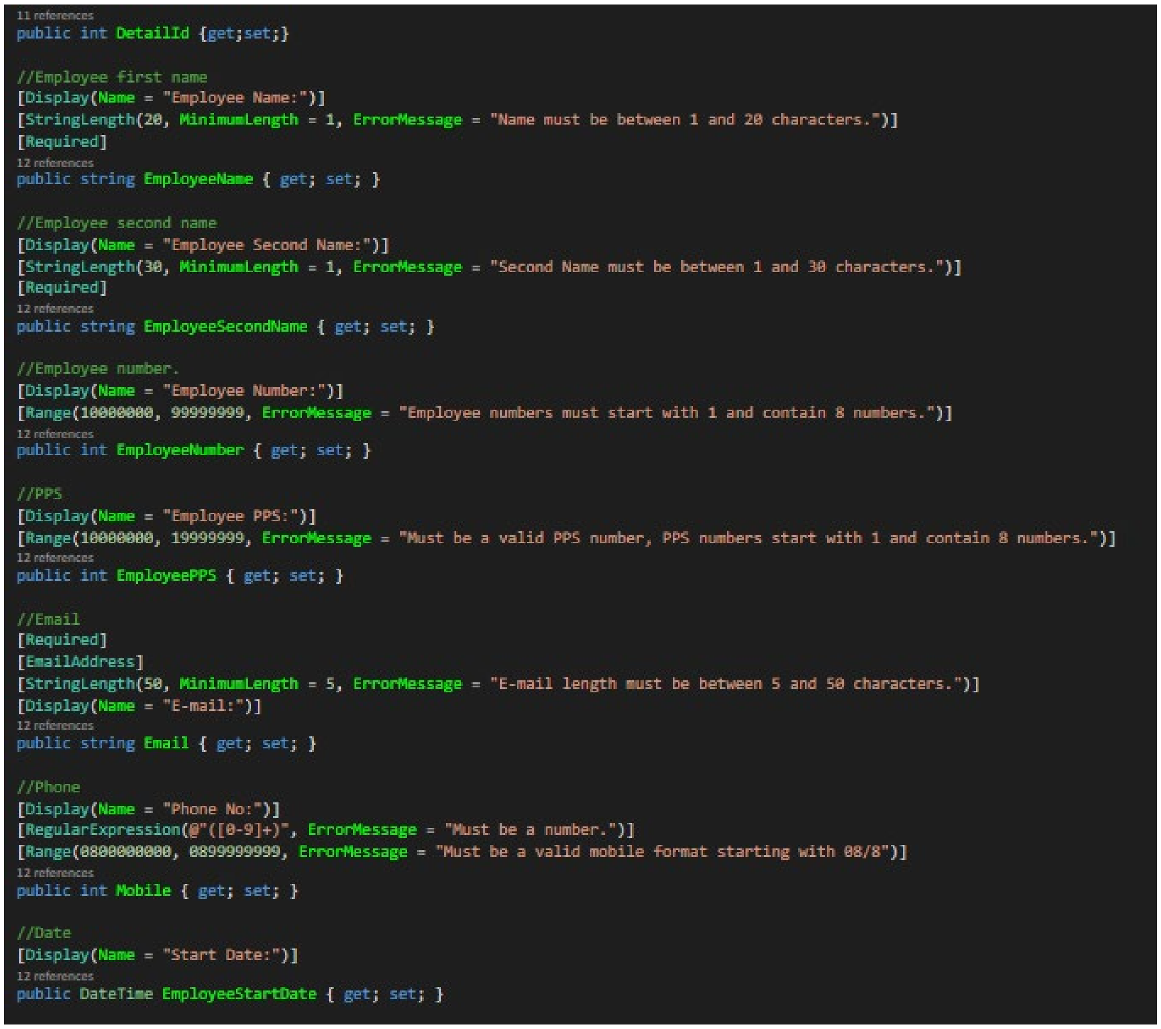
2.4 Security implementation

2.4.1 Secure coding

2.4.1.1 Input validation:

Input validations disallows the use of inappropriate entries in the data fields. This means that malicious scripts cannot be injected or wrong data is inserted into the fields.

AddDetail model:



```
11 references
public int DetailId {get;set;}

//Employee first name
[Display(Name = "Employee Name:")]
[StringLength(20, MinimumLength = 1, ErrorMessage = "Name must be between 1 and 20 characters.")]
[Required]
12 references
public string EmployeeName { get; set; }

//Employee second name
[Display(Name = "Employee Second Name:")]
[StringLength(30, MinimumLength = 1, ErrorMessage = "Second Name must be between 1 and 30 characters.")]
[Required]
12 references
public string EmployeeSecondName { get; set; }

//Employee number.
[Display(Name = "Employee Number:")]
[Range(10000000, 99999999, ErrorMessage = "Employee numbers must start with 1 and contain 8 numbers.")]
12 references
public int EmployeeNumber { get; set; }

//PPS
[Display(Name = "Employee PPS:")]
[Range(10000000, 19999999, ErrorMessage = "Must be a valid PPS number, PPS numbers start with 1 and contain 8 numbers.")]
12 references
public int EmployeePPS { get; set; }

//Email
[Required]
[EmailAddress]
[StringLength(50, MinimumLength = 5, ErrorMessage = "E-mail length must be between 5 and 50 characters.")]
[Display(Name = "E-mail:")]
12 references
public string Email { get; set; }

//Phone
[Display(Name = "Phone No:")]
[RegularExpression(@"([0-9]+)", ErrorMessage = "Must be a number.")]
[Range(0800000000, 0899999999, ErrorMessage = "Must be a valid mobile format starting with 08/8")]
12 references
public int Mobile { get; set; }

//Date
[Display(Name = "Start Date:")]
12 references
public DateTime EmployeeStartDate { get; set; }
```

It can be seen from the source code that all the fields have some kind of input validation, depending on the requirements.

Required fields are denoted with [Required] annotation. Integers and dates are automatically required without explicit [Required] annotation.

For the **AddDetail** model, the following are descriptions of how each variable is validated.

First name: must be minimum of 1 character and maximum 20.
Second name: must be minimum of 1 character and maximum 30.
Employee number: must contain 8 numbers.

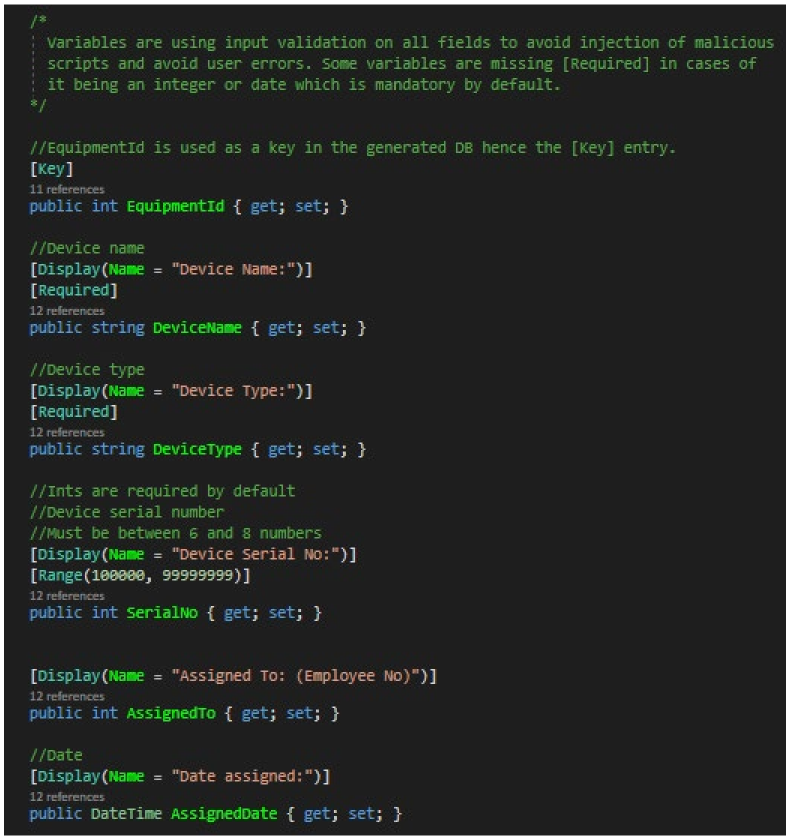
Employee PPS: must follow a valid format which starts with 1 and contains 8 numbers.

Email: email entered must be in the correct format (test@gmail.com)

Phone: phone numbers must follow a set standard which is starting with 08 and contain 10 numbers.

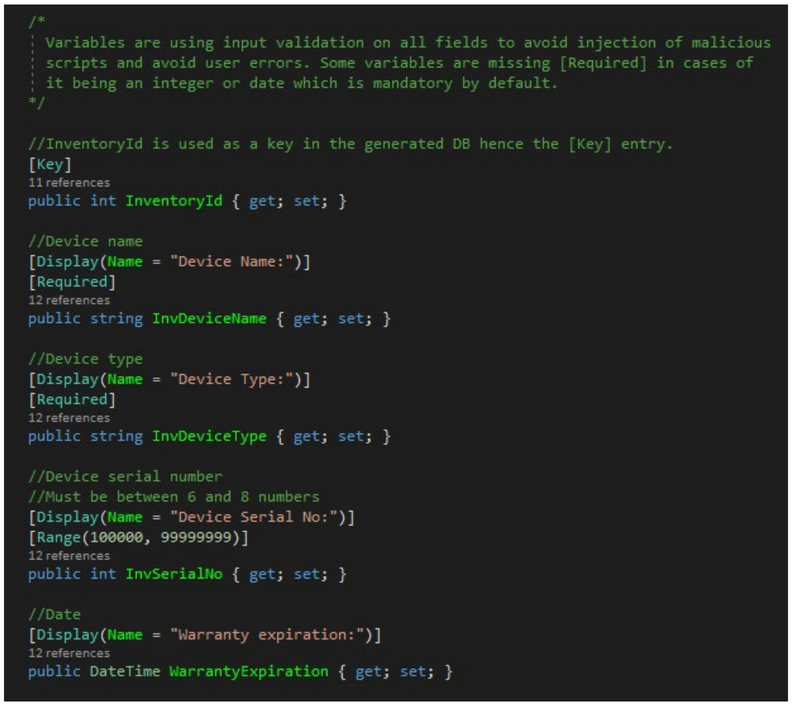
Date: date field can be selected using a table or entered manually by following the provided format.

AddEquipment model:



```
/*
 Variables are using input validation on all fields to avoid injection of malicious
 scripts and avoid user errors. Some variables are missing [Required] in cases of
 it being an integer or date which is mandatory by default.
*/

//EquipmentId is used as a key in the generated DB hence the [Key] entry.
[Key]
11 references
public int EquipmentId { get; set; }

//Device name
[Display(Name = "Device Name:")]
[Required]
12 references
public string DeviceName { get; set; }

//Device type
[Display(Name = "Device Type:")]
[Required]
12 references
public string DeviceType { get; set; }

//Ints are required by default
//Device serial number
//Must be between 6 and 8 numbers
[Display(Name = "Device Serial No:")]
[Range(100000, 99999999)]
12 references
public int SerialNo { get; set; }


[Display(Name = "Assigned To: (Employee No)")]
12 references
public int AssignedTo { get; set; }

//Date
[Display(Name = "Date assigned:")]
12 references
public DateTime AssignedDate { get; set; }
```

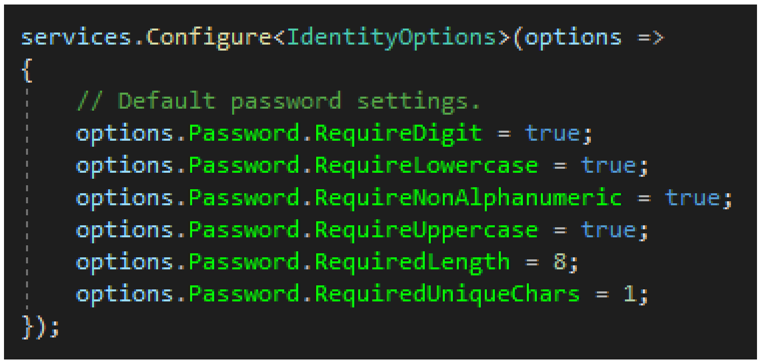
Inventory model:



```
/*
 Variables are using input validation on all fields to avoid injection of malicious
 scripts and avoid user errors. Some variables are missing [Required] in cases of
 it being an integer or date which is mandatory by default.
*/

//InventoryId is used as a key in the generated DB hence the [Key] entry.
[Key]
11 references
public int InventoryId { get; set; }

//Device name
[Display(Name = "Device Name:")]
[Required]
12 references
public string InvDeviceName { get; set; }

//Device type
[Display(Name = "Device Type:")]
[Required]
12 references
public string InvDeviceType { get; set; }

//Device serial number
//Must be between 6 and 8 numbers
[Display(Name = "Device Serial No:")]
[Range(100000, 99999999)]
12 references
public int InvSerialNo { get; set; }

//Date
[Display(Name = "Warranty expiration:")]
12 references
public DateTime WarrantyExpiration { get; set; }
```

Password validation:

Password rules have been changed to require a digit, lowercase character, uppercase character, non-alphanumeric character, at least 1 unique character and have a minimum length of 8 characters. This is done to ensure no weak passwords are used and the application is not susceptible to password guessing attacks.



```
services.Configure<IdentityOptions>(options =>
{
    // Default password settings.
    options.Password.RequireDigit = true;
    options.Password.RequireLowercase = true;
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequireUppercase = true;
    options.Password.RequiredLength = 8;
    options.Password.RequiredUniqueChars = 1;
});
```

This is done by configuring the IdentityOptions service with desired options.

Input validation and password requirements (HTML):

Input validation can also be done using HTML instead of or in addition to C#. This is done by using the following code for a password:

<input type="**password**" id="password" name="confirm_password" class="form-control" placeholder="Enter your desired password" **maxlength="20" pattern="(?=.*\d)(?=.*[!@#$%^&*])(?=.*[a-z])(?=.*[A-Z]).{8,}"** title="Password must contain at least one uppercase letter, one number, one special character and be at least 8 characters long."

This enforces password strength which prevents users entering a weak password. Size limits are also added to ensure that no long scripts are entered.

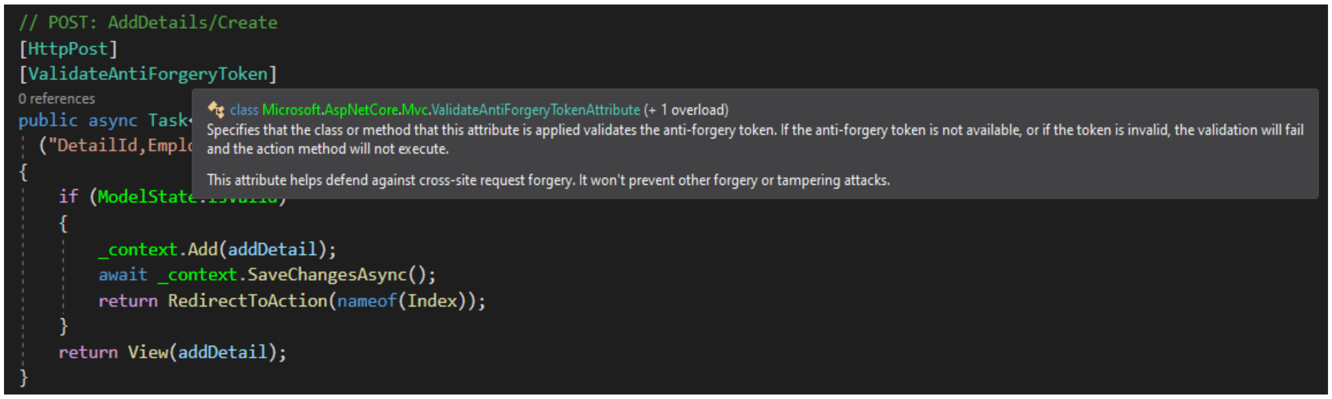
<input **type="email"** id="email" name="email" class="form-control" placeholder="E.g. johsmith@gmail.com" **maxlength="40"**

This allows only e-mail format messages to be inputted. Any other input gets rejected.

2.4.1.2 ValidateAntiForgeryToken()

One advantage of using ASP.NET Core is the implementation of secure features. The attribute “ValidateAntiForgeryToken” is added to prevent cross-site request forgery attacks. It is part of the ASP.NET Core tools.

ValidateAntiForgeryToken works by generating a different token for every request made which eliminates request forging.



```
// POST: AddDetails/Create
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task
  ("DetailId,Emplo
{
    if (ModelState
    {
        _context.Add(addDetail);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(addDetail);
}
```

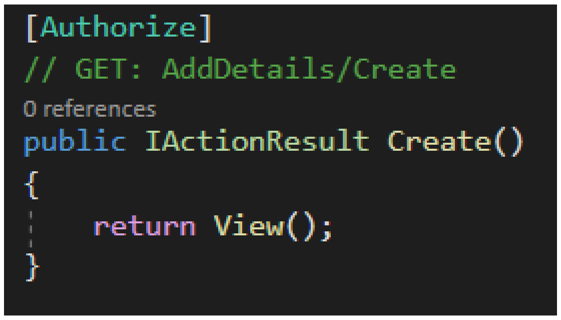
class Microsoft.AspNetCore.Mvc.ValidateAntiForgeryTokenAttribute (+ 1 overload)
Specifies that the class or method that this attribute is applied validates the anti-forgery token. If the anti-forgery token is not available, or if the token is invalid, the validation will fail and the action method will not execute.

This attribute helps defend against cross-site request forgery. It won't prevent other forgery or tampering attacks.

2.4.1.3 Authorization

The application only allows changes to be made by users who have an account and are logged into the application. Users that are not logged in, are redirected to the login page where they can either login to their existing account or go to the registration page to create a new account. This requirement can be customized depending on the areas which require authorization. In this project, any create, update or delete functions require user to be logged in. This is completed using [Authorize] attribute applied to the relevant action.

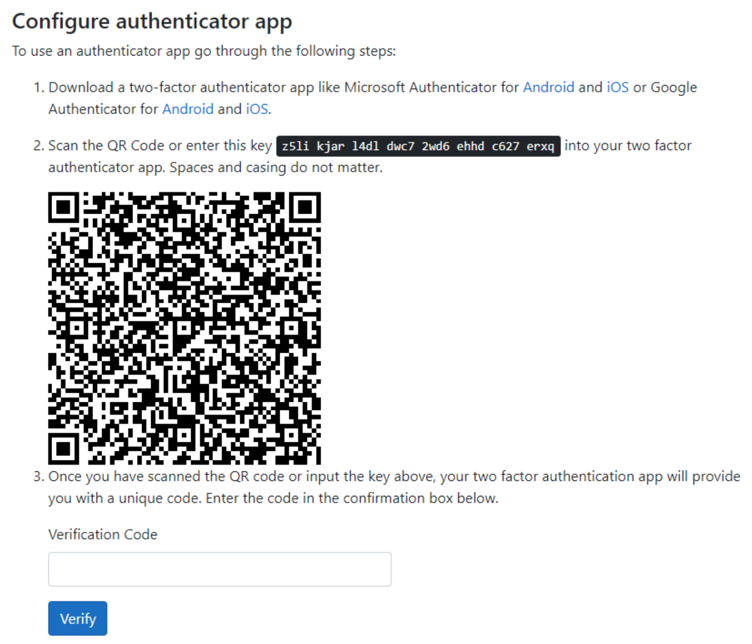
Login requirement applied to returning the page for AddDetails view.



```
[Authorize]
// GET: AddDetails/Create
0 references
public IActionResult Create()
{
    return View();
}
```

2.4.1.4 Multi factor authentication:

Multi factor authentication is a great tool for ensuring only authorized parties are accessing the application. Existing users with 2 factor authentication enabled, are required to confirm their logins by entering their code using an authenticator app on their mobile. This prevents malicious user logins, as every login attempt has to be verified by the user themselves.



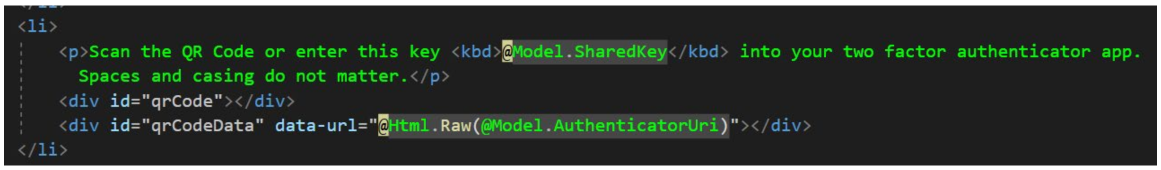
Configure authenticator app

To use an authenticator app go through the following steps:

1. Download a two-factor authenticator app like Microsoft Authenticator for Android and iOS or Google Authenticator for Android and iOS.
2. Scan the QR Code or enter this key `z5li kjar 14dl dwc7 2wd6 ehhd c627 erxq` into your two factor authenticator app. Spaces and casing do not matter.
3. Once you have scanned the QR code or input the key above, your two factor authentication app will provide you with a unique code. Enter the code in the confirmation box below.

Verification Code

A shared key is generated and displayed to the user:



```
<li>
    <p>Scan the QR Code or enter this key <kbd>@Model.SharedKey</kbd> into your two factor authenticator app.
      Spaces and casing do not matter.</p>
    <div id="qrCode"></div>
    <div id="qrCodeData" data-url="@Html.Raw(@Model.AuthenticatorUri)"></div>
</li>
```

A public library is used to provide QR code option which makes it easier to enable 2 factor authentication by scanning the QR code which automatically adds the application to the authenticator app on the phone. If using Microsoft Authenticator, the codes are renewed every 30 seconds.

2.4.1.5 Preventing injections using Entity Framework.

Entity framework is an object-relational mapping framework.

Using code-first approach, database is generated by the ORM. This removes the focus on the code that handles databases. This indirect approach to database management prevents alterations that are made to the database.

Injections are bypassed by the use of parameterized queries therefore no SQL injection is possible by using Entity Framework. No SQL queries are included in the application.

//Not likely to use any PHP, just in case it is required.

2.4.1.6 Sanitizing user input in PHP

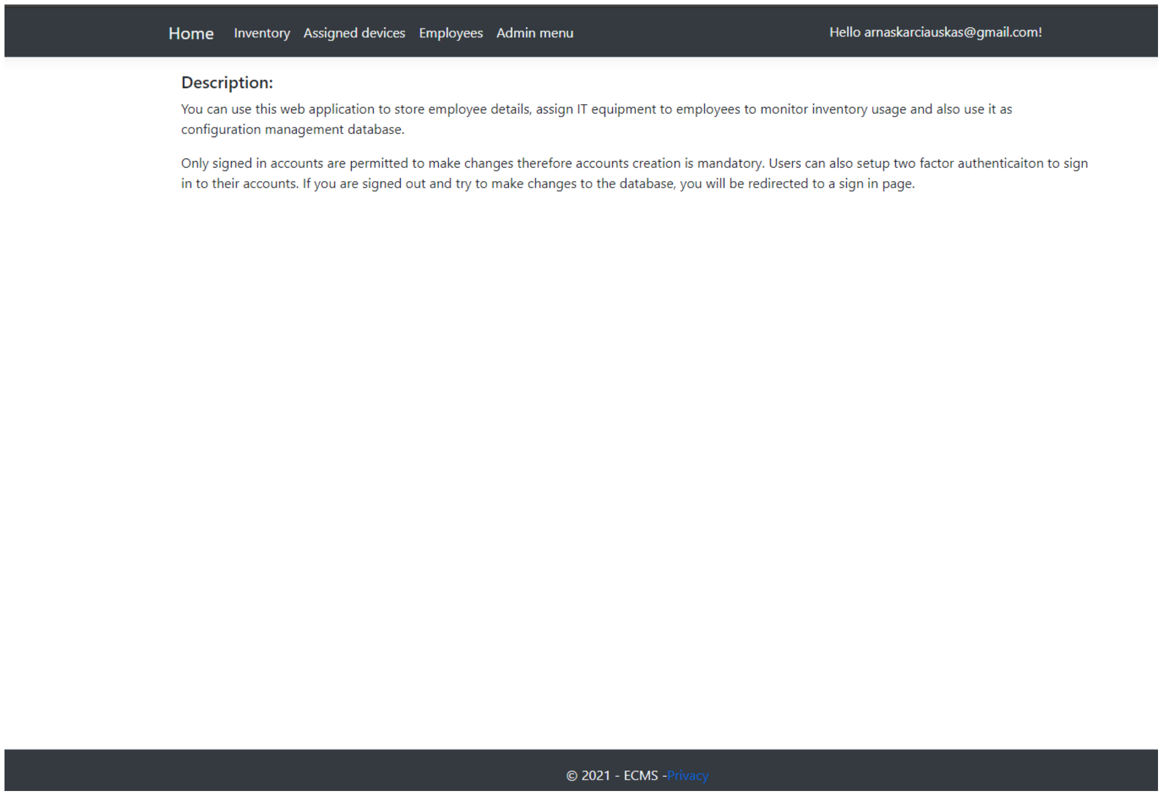
```
function sanitize(s) {

return s.replace(/&/g, '&amp;').replace(/</g, '&lt;').replace(/"/g, '&quot;');

}
```

This ensures that any scripts tags become $lt; and therefore does not run any tags with scripts tags.

2.5 Graphical User Interface (GUI) Layout

Index page:

Index page displays basic application information and its features.



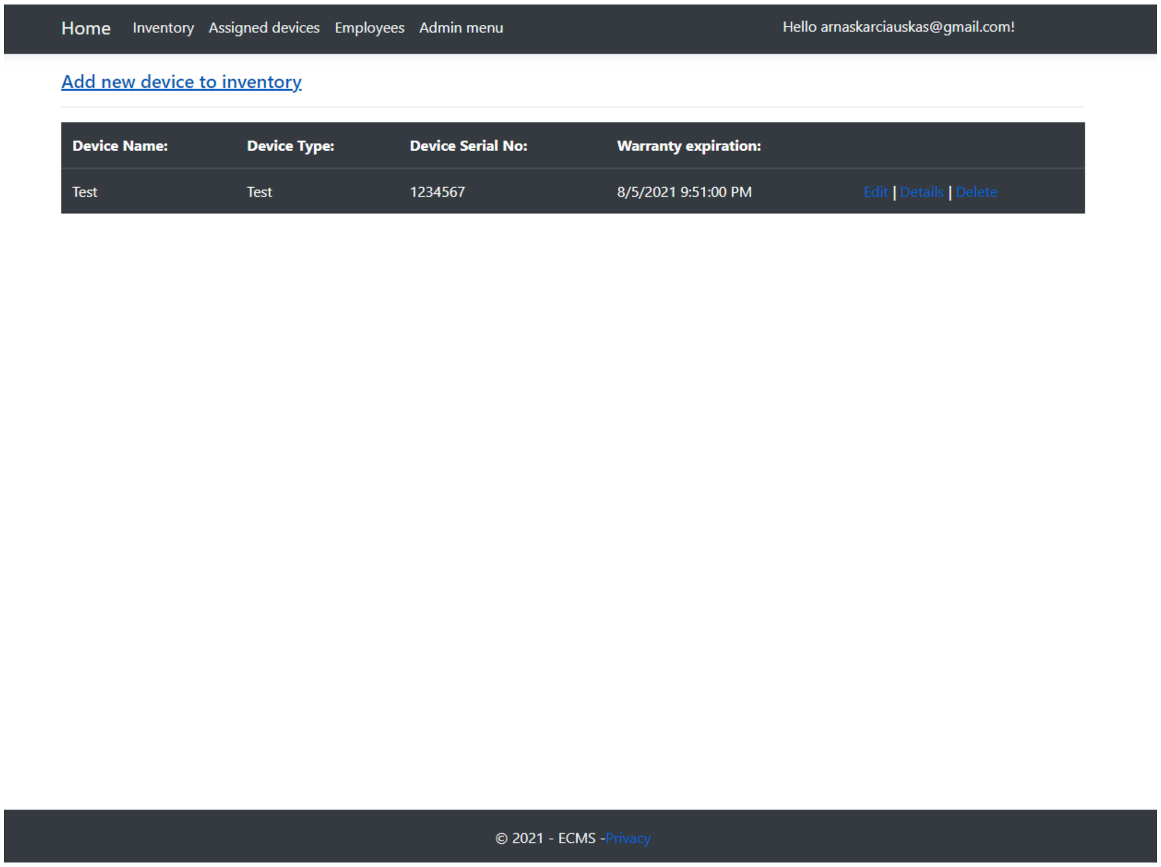
Description:

You can use this web application to store employee details, assign IT equipment to employees to monitor inventory usage and also use it as configuration management database.

Only signed in accounts are permitted to make changes therefore accounts creation is mandatory. Users can also setup two factor authenticaiton to sign in to their accounts. If you are signed out and try to make changes to the database, you will be redirected to a sign in page.

Inventory page:

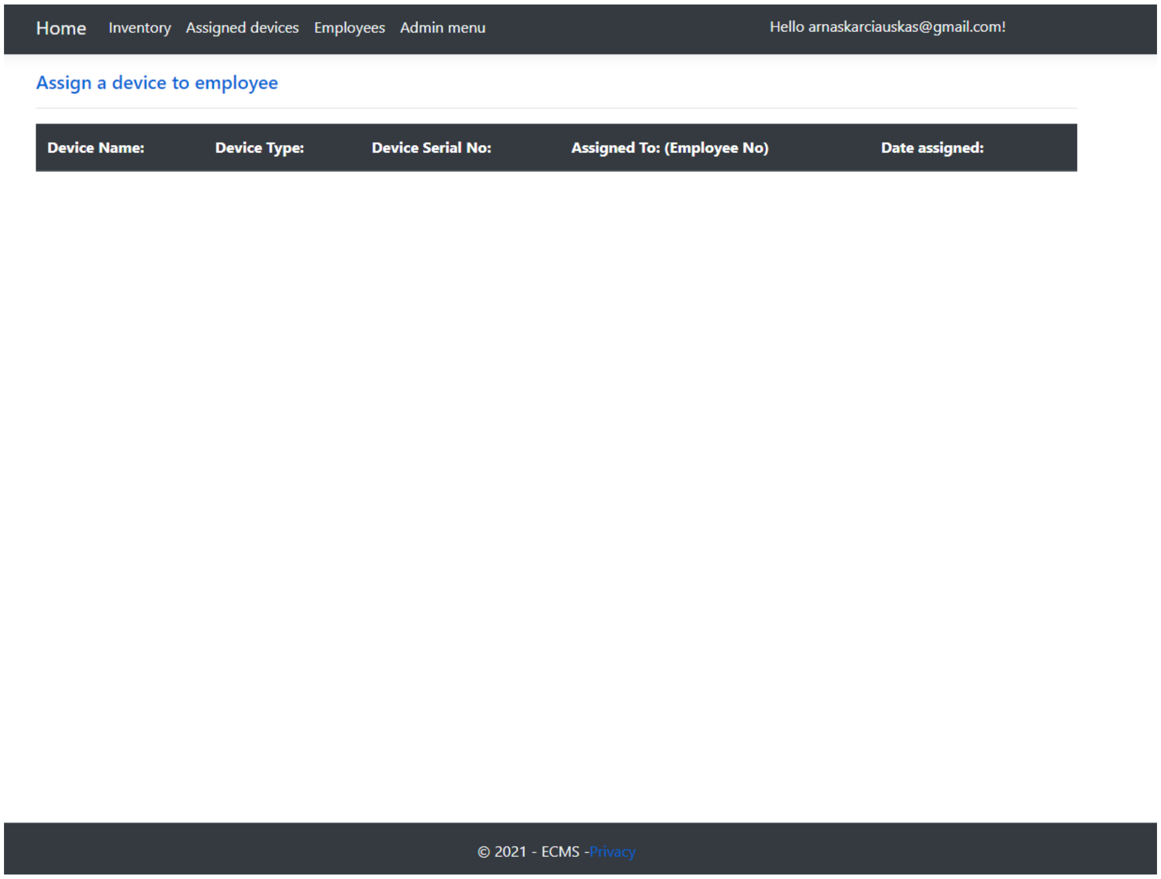
Authorized users can view existing entries in the inventory database. Authorized users can also perform create, read, update and delete functions from this screen.


Home
Inventory
Assigned devices
Employees
Admin menu
Hello arnaskarciauskas@gmail.com!
Add new device to inventory

Device Name:	Device Type:	Device Serial No:	Warranty expiration:	
Test	Test	1234567	8/5/2021 9:51:00 PM	Edit | Details | Delete

Device page:

Authorized users can view assigned entries database. Authorized users can also perform create, read, update and delete functions from this screen.

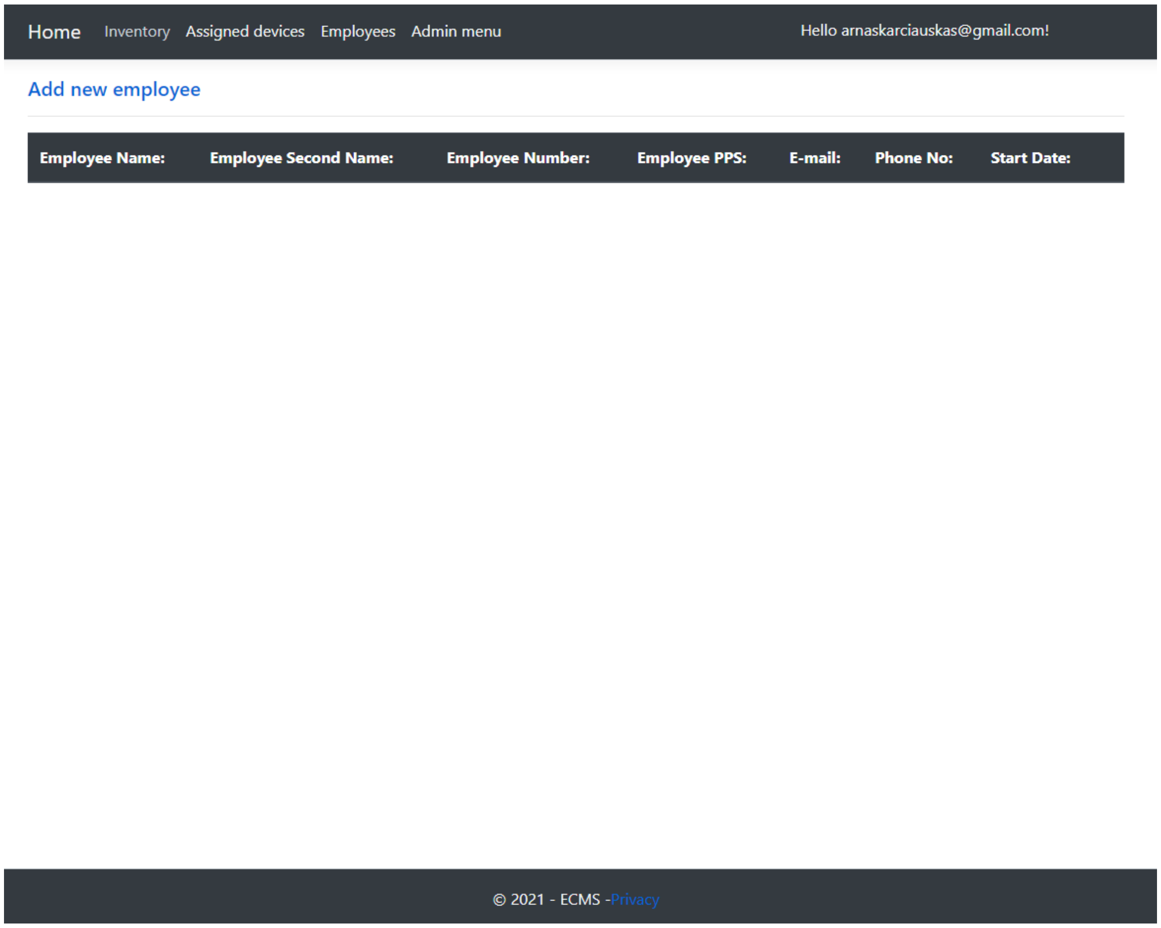

Home
Inventory
Assigned devices
Employees
Admin menu
Hello arnaskarciauskas@gmail.com!
Assign a device to employee

Device Name:	Device Type:	Device Serial No:	Assigned To: (Employee No)	Date assigned:

© 2021 - ECMS -Privacy

Employees page:

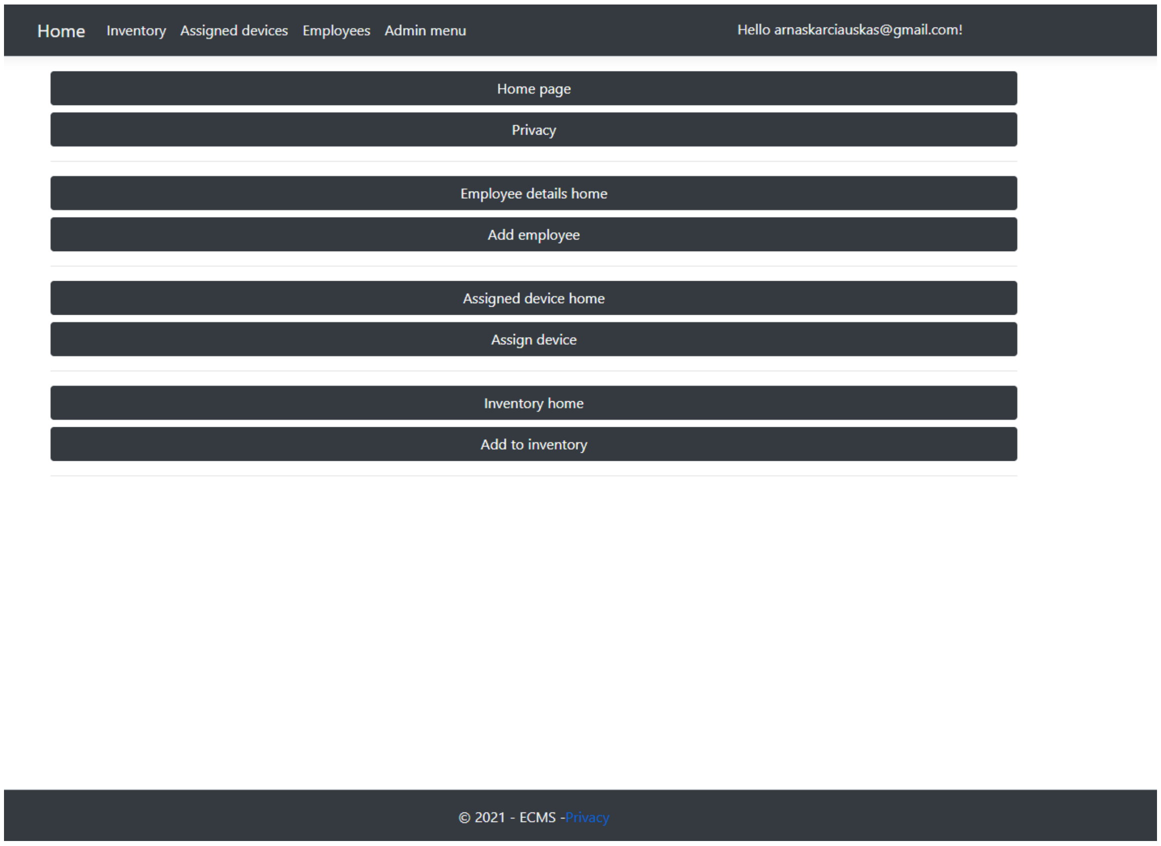
Authorized users can view employee entries database. Authorized users can also perform create, read, update and delete functions from this screen.



Employee Name:	Employee Second Name:	Employee Number:	Employee PPS:	E-mail:	Phone No:	Start Date:

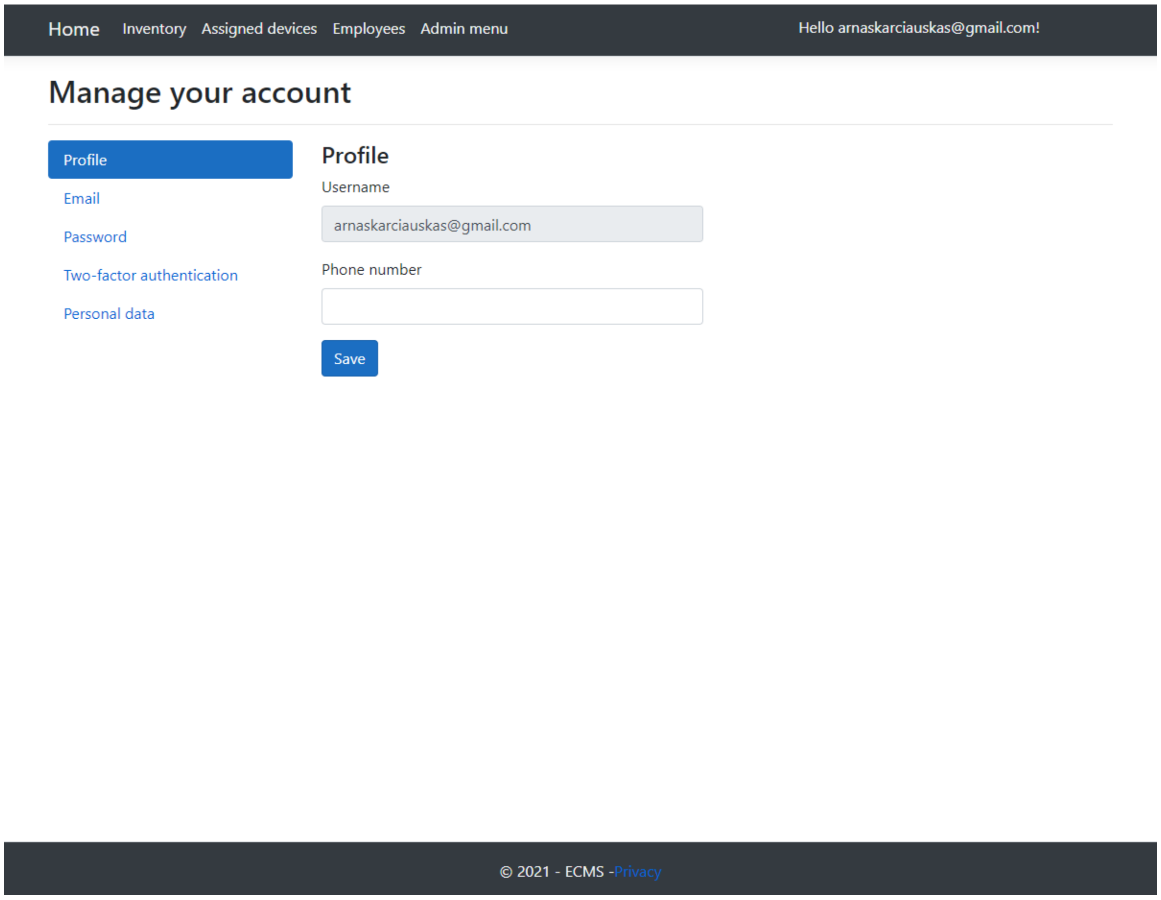
Admin menu:

Authorized users can visit home pages of all models and add new entries to the database from this menu. If users want to make changes to existing entries, they would need to go to the relevant section main page.



Profile management page:

Registered users can change their account details from this screen. Options available are: phone number, email, password two-factor authentication and personal data.



2.6 Testing

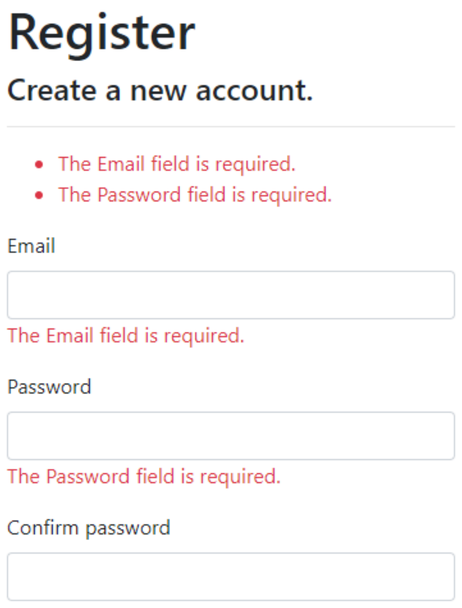
2.6.1 Security testing:

The web application will be security tested to discover any threats to the web application. Testing will be performed for injection and input validation to ensure there are no vulnerabilities related to these areas.

The application is tested by trying to circumvent the rules in place.

Input validation:

Trying to register an account with empty fields:



Register

Create a new account.

- The Email field is required.
- The Password field is required.

Email

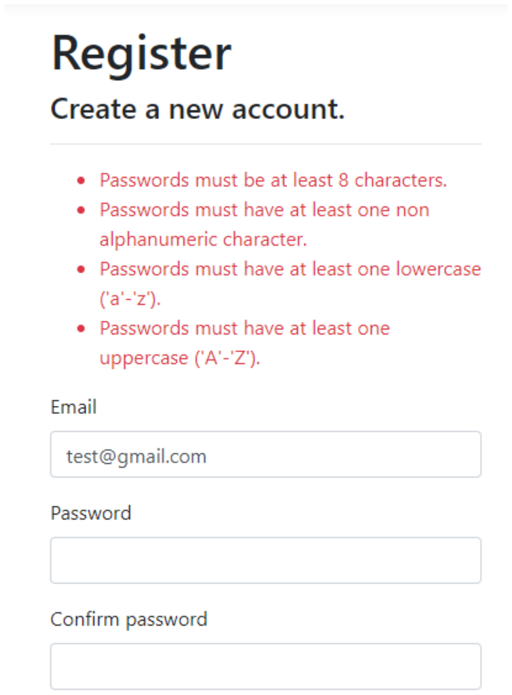
The Email field is required.

Password

The Password field is required.

Confirm password

Trying to create an account with a weak password:



Register

Create a new account.

- Passwords must be at least 8 characters.
- Passwords must have at least one non alphanumeric character.
- Passwords must have at least one lowercase ('a'-'z').
- Passwords must have at least one uppercase ('A'-'Z').

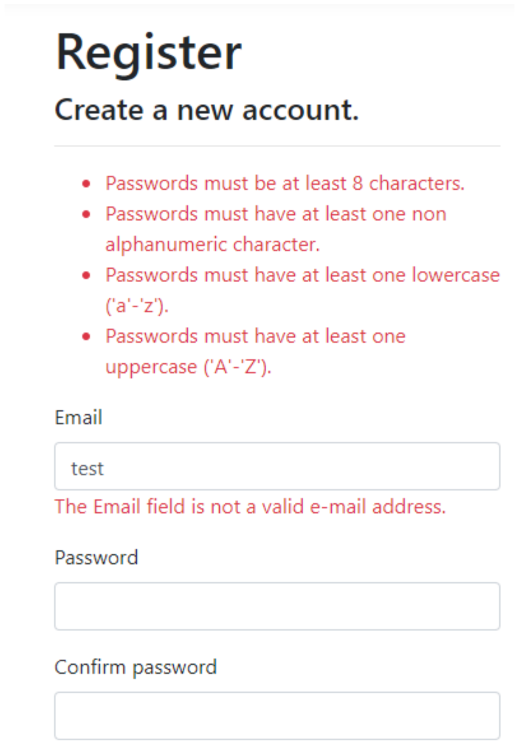
Email

test@gmail.com

Password

Confirm password

Trying to create an account with incorrect format:



Register

Create a new account.

- Passwords must be at least 8 characters.
- Passwords must have at least one non alphanumeric character.
- Passwords must have at least one lowercase ('a'-'z').
- Passwords must have at least one uppercase ('A'-'Z').

Email

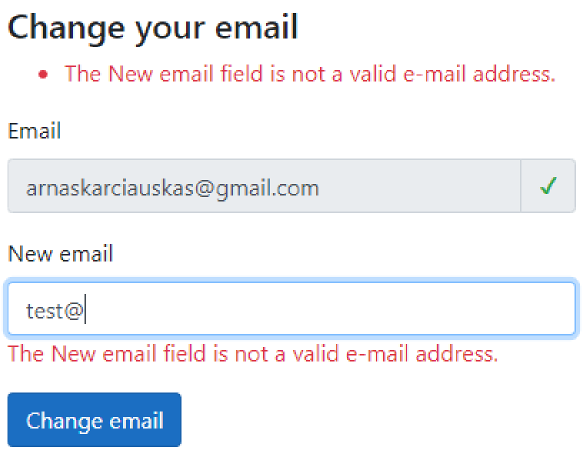
test

The Email field is not a valid e-mail address.

Password

Confirm password

Trying to change an email to incorrect format:


Change your email

- The New email field is not a valid e-mail address.

Email

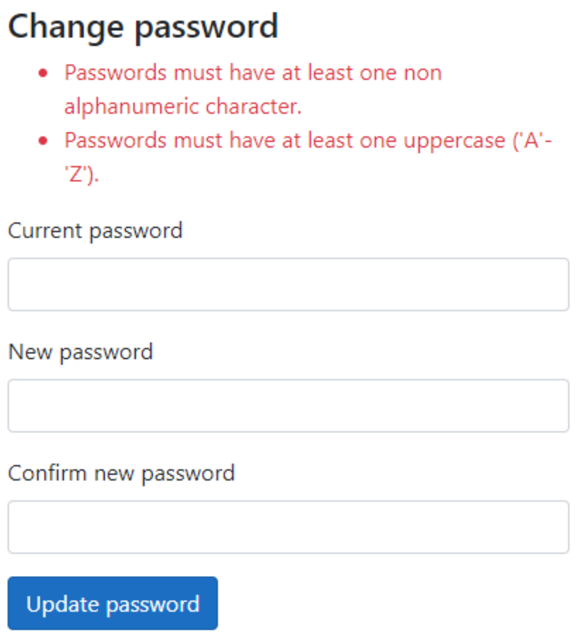
arnaskarciauskas@gmail.com ✓

New email

test@

The New email field is not a valid e-mail address.

Changing existing password:


Change password

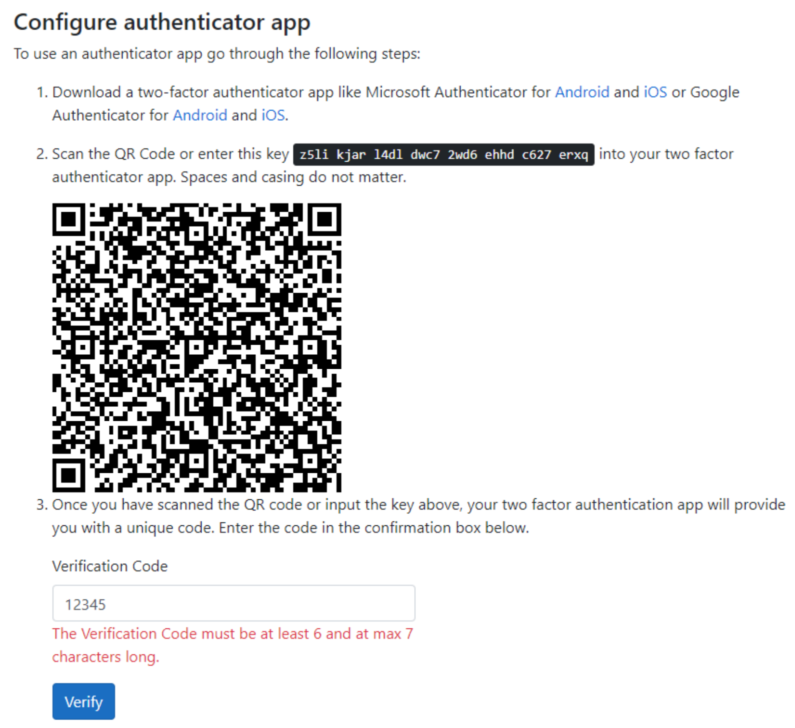
- Passwords must have at least one non alphanumeric character.
- Passwords must have at least one uppercase ('A'-'Z').

Current password

New password

Confirm new password

Trying to enter an invalid format for 2-factor authentication code.



Configure authenticator app

To use an authenticator app go through the following steps:

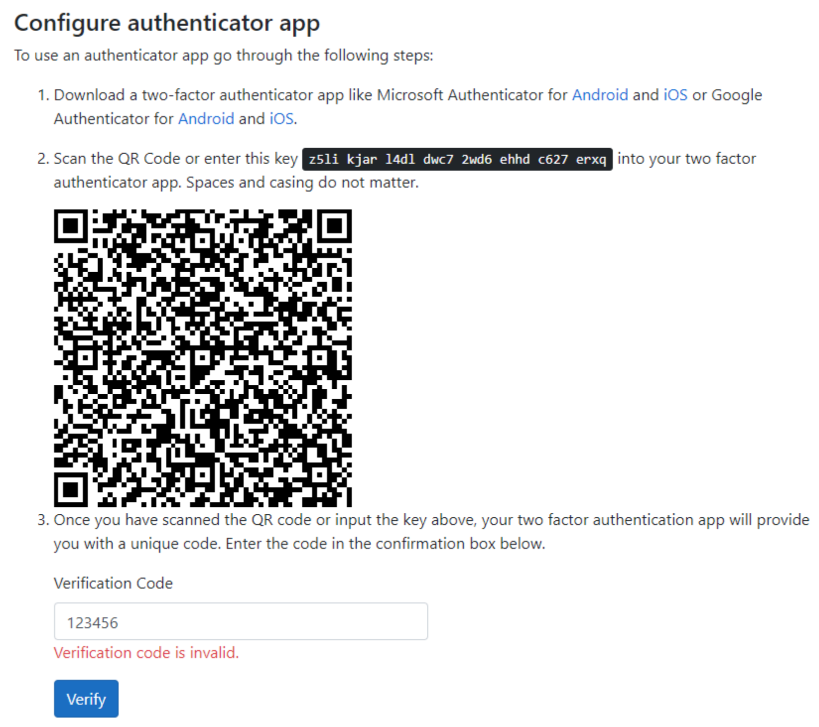
1. Download a two-factor authenticator app like Microsoft Authenticator for Android and iOS or Google Authenticator for Android and iOS.
2. Scan the QR Code or enter this key `z5li kjar 14dl dwc7 2wd6 ehhd c627 erxq` into your two factor authenticator app. Spaces and casing do not matter.
3. Once you have scanned the QR code or input the key above, your two factor authentication app will provide you with a unique code. Enter the code in the confirmation box below.

Verification Code

12345

The Verification Code must be at least 6 and at max 7 characters long.

Entering a different code as to what is provided by the application to setup the 2-factor authentication.



Configure authenticator app

To use an authenticator app go through the following steps:

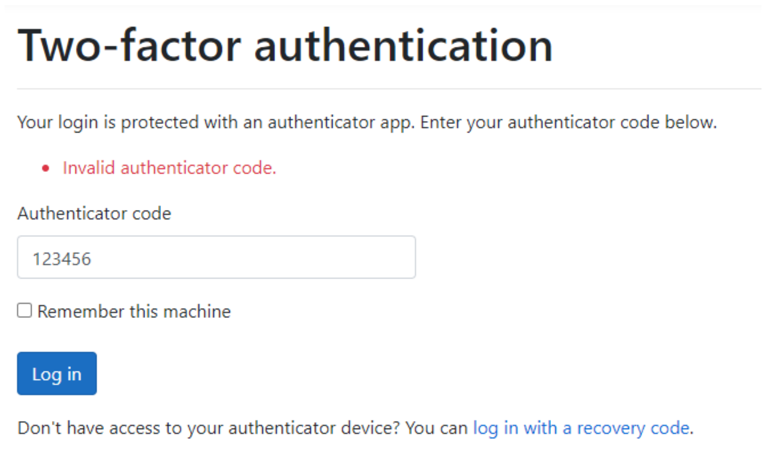
1. Download a two-factor authenticator app like Microsoft Authenticator for Android and iOS or Google Authenticator for Android and iOS.
2. Scan the QR Code or enter this key `z5li kjar 14dl dwc7 2wd6 ehhd c627 erxq` into your two factor authenticator app. Spaces and casing do not matter.
3. Once you have scanned the QR code or input the key above, your two factor authentication app will provide you with a unique code. Enter the code in the confirmation box below.

Verification Code

123456

Verification code is invalid.

Inputting a different to as to what is provided by the Microsoft authenticator application:



Two-factor authentication

Your login is protected with an authenticator app. Enter your authenticator code below.

- Invalid authenticator code.

Authenticator code

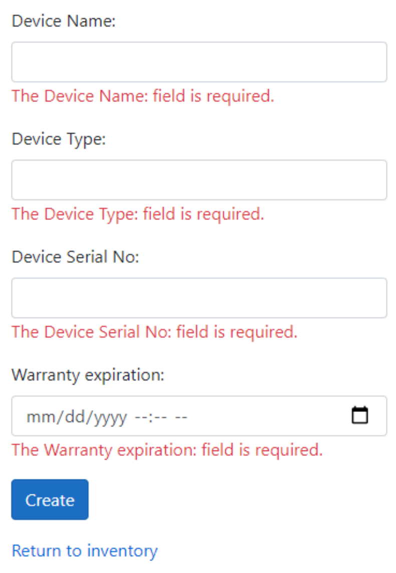
123456

☐ Remember this machine

Log in

Don't have access to your authenticator device? You can log in with a recovery code.

Trying to create a new empty inventory entry:



Device Name:

The Device Name: field is required.

Device Type:

The Device Type: field is required.

Device Serial No:

The Device Serial No: field is required.

Warranty expiration:

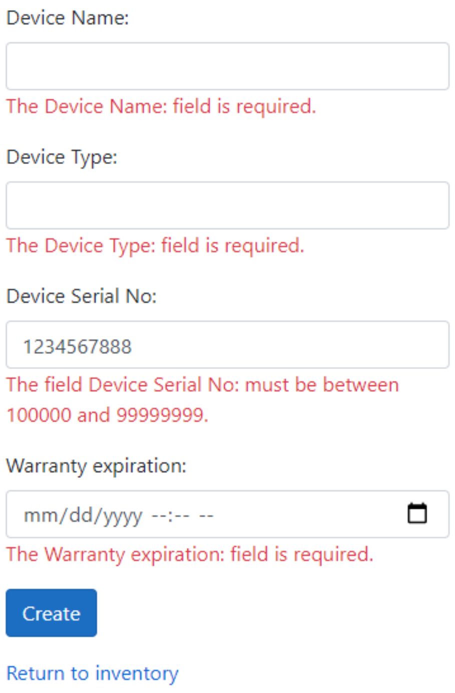
mm/dd/yyyy --:-- --

The Warranty expiration: field is required.

Create

Return to inventory

Trying to create a new inventory entry with incorrect format:


Device Name:

The Device Name: field is required.

Device Type:

The Device Type: field is required.

Device Serial No:

1234567888

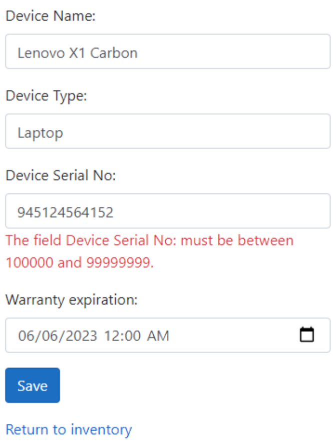
The field Device Serial No: must be between 100000 and 99999999.

Warranty expiration:

mm/dd/yyyy --:-- --

The Warranty expiration: field is required.

Trying to change an existing entry with incorrect details:


Device Name:

Lenovo X1 Carbon

Device Type:

Laptop

Device Serial No:

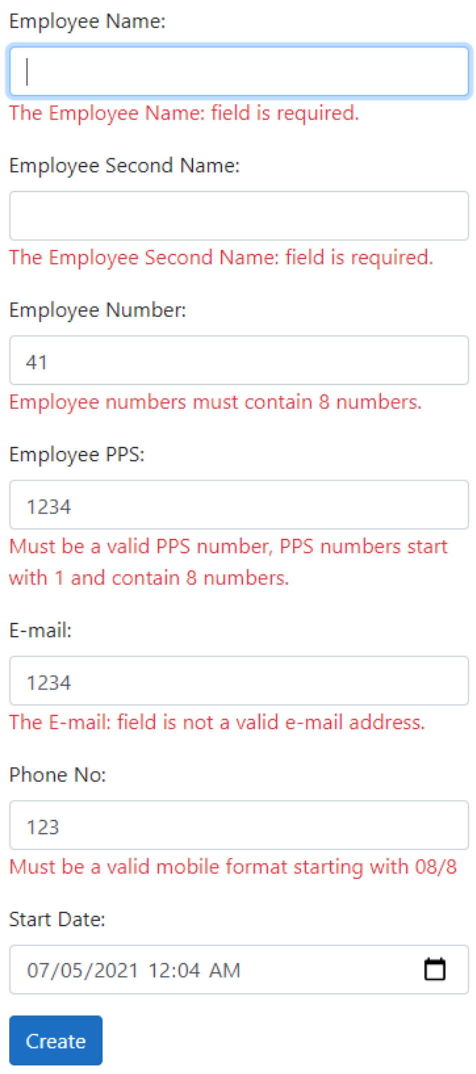
945124564152

The field Device Serial No: must be between 100000 and 99999999.

Warranty expiration:

06/06/2023 12:00 AM

Input validation for adding employee, trying to explicitly enter incorrect entries, this helps with user error and prevents malicious attempts:



Employee Name:

The Employee Name: field is required.

Employee Second Name:

The Employee Second Name: field is required.

Employee Number:

41

Employee numbers must contain 8 numbers.

Employee PPS:

1234

Must be a valid PPS number, PPS numbers start with 1 and contain 8 numbers.

E-mail:

1234

The E-mail: field is not a valid e-mail address.

Phone No:

123

Must be a valid mobile format starting with 08/8

Start Date:

07/05/2021 12:04 AM

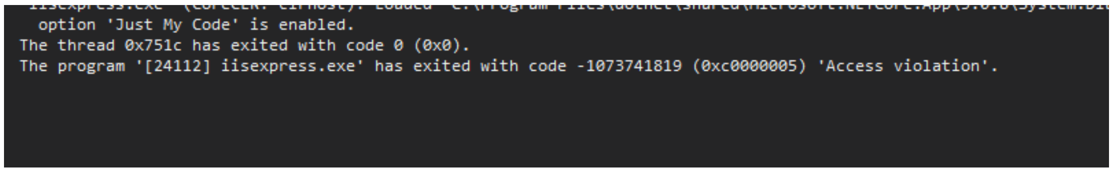
2.6.2 Application testing:

All parts of the application will be tested to ensure that they function correctly. Registration system, user details change, account deletion etc. All parts of the web application will be extensively tested and populated with data to ensure consistency.

Test results:

Operation	**Expected Result**	**Actual Result**	**Overall result**
Launching application	Open successfully without errors	Working	Pass
Registering account	Account registered with no errors	Working	Pass
Login	Login with existing account	Working	Pass
2 Factor authentication	Setup 2 factor authentication successfully	Working	Pass
Login using 2FA	Successful login by inputting code from authentication app	Working	Pass
Add phone number	Successfully add a mobile number on personal account	Working	Pass
Change email	Change existing email to a different one	Working	Pass
Change password	Change existing password to a new one	Working	Pass
Download details	Successfully download all account details	Working	Pass
Delete account	Account is deleted successfully	Working	Pass
Perform create operation	**Create at least 1 new entry on all models**	**Fail**	**Fail**
Perform delete operation	Perform delete operation on at least 1 entry on all models	Working	Pass
Perform edit operation	Perform edit operation on at least 1 entry on all models	Working	Pass

Application has performed majority of the steps successfully, however it has failed the operation “Perform create operation” on “AddEquipment” model. This was caused by unusued ICollection method, which was removed in order to resolve the root cause. Tests was rerun and ran successfully after the code change.



```
  option 'Just My Code' is enabled.
The thread 0x751c has exited with code 0 (0x0).
The program '[24112] iisexpress.exe' has exited with code -1073741819 (0xc0000005) 'Access violation'.
```

Perform create operation	Create at least 1 new entry on all models	Working	Pass

3 Conclusions

This project aims to address a gap in the market in relation to data management in organizations. While it provides limited functionality, it has the potential to be developed further to increase functionality of the web application to apply more to the mass market. The disadvantage of this application is that it caters to small organizations and is not a mass market product at the current level of development. It would require further development and added functionality to be able to be marketed to larger organizations which need extensive functionality and certification of conformity. A larger team with various specializations would be required to bring the application to another level.

4 Further development or research

With more resources, this project would be marketed to larger businesses, which deal with a high amount of assets and employees. Organizations could use this application to maintain records of all the assets and employee information. The application would also be split into 2 different apps, one to be a standalone IT configuration management database and another one HR management tool with extra functionality such as clock in times for staff. The focus would be to further develop the project and sell it to larger organizations with support packages for an affordable price.

Additional features to be added in case of project expansion:

- Notifications sent to administrator in regards to asset warranty expirations.
- Notifications sent when unauthorized access is attempted.
- Extensive logging and transfer to a remote server of syslogs.
- Split the application into two different applications for IT configuration management and HR tool.
- Development of a mobile application so that users could have easy access through their phones. Data uploaded offline to be synced when the user gains an internet connection.

5 References

Docs.microsoft.com. (2019). *Overview of ASP.NET Core MVC*. [online] Available at: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-3.1 [Accessed 5 Dec. 2019].

Owasp.org. (2019). [online] Available at: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf [Accessed 4 Dec. 2020].

Owasp.org. 2021. [online] Available at: https://owasp.org/www-pdf-archive/OWASP_Top_10_-_2013.pdf [Accessed 1 August 2021].

C-sharpcorner.com. (2019). *Upload Files In ASP.NET MVC 5*. [online] Available at: https://www.c-sharpcorner.com/article/upload-files-in-asp-net-mvc-5 [Accessed 7 Dec. 2020].

Docs.microsoft.com. (2019). *Security Considerations (Entity Framework)*. [online] Available at: https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-4.0/cc716760(v=vs.100)?redirectedfrom=MSDN [Accessed 10 May. 2021].

Bu.edu. (2019). *Understanding Authentication, Authorization, and Encryption : TechWeb : Boston University*. [online] Available at: http://www.bu.edu/tech/about/security-resources/bestpractice/auth/ [Accessed 10 May. 2021].

Morgan, D. 2006, "Web application security - SQL injection attacks", *Network Security,* vol. 2006, no. 4, pp. 4-5.

Davidshimjs.github.io. 2021. *qrcode.js*. [online] Available at: https://davidshimjs.github.io/qrcodejs/ [Accessed 31 July 2021].

Docs.microsoft.com. 2021. *Testing ASP.NET Core services and web apps*. [online] Available at: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/test-aspnet-core-services-web-apps [Accessed 1 August 2021].

Docs.microsoft.com. 2021. *Introduction to ASP.NET Identity - ASP.NET 4.x*. [online] Available at: https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity [Accessed 1 August 2021].

6 Appendix

7 Proposal

7.1 What will the project do?

The project is a web application that allows users to manage business data which includes employee data and IT inventory. It allows management of assets and the ability to assign them to employees for tracking purposes.

7.2 Main functionality:

- Managing employee data
- Managing company assets
- Improve record keeping and ease of use
- Easy to use
- Emphasis on security
- Controlled by administrator using 2 factor authentication
- Ability for administrators to create, read, update and delete records

7.3 User features:

1. User management system (Registration, login)
2. Personal data download or delete
3. Change password
4. Change email
5. 2 factor authentication
6. Adding phone number

7.4 Administration:

1. Create, read update and delete of data
2. See information about entries
3. Make changes to layout

7.5 Why is it challenging?

The reason that the project is challenging is because there are many aspects of the web development which I have no knowledge or skill in. That is details such as:

- Advanced security functionality for protection of web applications.
- Functional registration and login system.
- Multi factor authentication.
- Coding based on secure principles.

7.6 Why should this project be attempted?

This project should be attempted in order to gain new skills and fine tune existing ones to help create better secure web applications and improve overall security knowledge on coding principles as this newly gained knowledge will provide benefit to my career.

My main goal is to improve cyber security skills to advance skills and be able to create secure apps and provide guidance on creation of secure applications. By doing various tests and checking to see what works and what doesn't, it will help me to be able to implement security features in my full time job as part of the infrastructure operations team.

7.7 Reason for choosing a web application:

Throughout the 4 college years, my highest grades were always in modules related to web development and web application development. Rather than spending majority of the time learning new coding concepts to develop the application and spending a large amount of time troubleshooting which can take up a large portion of the time, I have decided to create a web application using C# ASP.Net Core framework. I learned about this technology in one of the modules in 3rd year which has shown to me the capabilities of the application. It showed the extensive security features of the framework which was ideal for my cyber security specialization.

This was reinforced by reading a journal from a 4th year student Justin Mulhern who has mentioned in his journal that he was struggling with coding by having to learn new concepts which he did not know before. This has made his final year project a big struggle as he had to spend so much time learning new concepts and troubleshooting new issues that have arose. I know that I have more skills and knowledge in web related development and security of web apps. Therefore it is my preferred choice as I am confident that I would do significantly better in such a project and have a better start instead of having to spend most of my time learning new concepts and trying to find solutions online for non-working code.

Objectives

The objective of this project is to provide an application used by employers to manage up to date company assets and employee data. The application also provides functionality for assigning devices to employees to keep track of the IT inventory functioning as a Configuration Management Database (CMDB), which is common in larger IT organizations.
Many operational companies and still do this the old way of keeping a paper record of the information and having issues keeping a good record of the data.

With the current advancements in technology, nearly every organization has a large amount of IT equipment required to perform daily business functions.

The application hopes to solve this problem by implementing a database of IT assets and employees. This would ensure easily accessible and easy to manage solution for an organization to manage their inventory and organizational data. The main goal is to make it more accessible and easy to use. The data can be backed up and easily found instead of being lost or stored away in some excel file which can be lost with the hard drive or a paper copy which can be accidentally lost in a pile of documents or be accidentally destroyed.

Goals of the project summarized:

- Managing employee database
- Managing IT asset database
- Easy to use
- Replacement for excel spreadsheets and paper copies
- Accessible anywhere with a network connection
- Protection against top vulnerabilities

8 Background

I have worked in many companies where the procedure of data management is almost exclusively based on paper or a single hard drive, meaning that the record can be lost or damaged. The reason for choosing this project is the potential to create this application which has an use in the real world scenario for small organizations which have no feasible method of managing their data. Commercial solutions usually have high pricing options which a smaller organization simply cannot afford or justify the expense.

I have researched for viable solutions online and could not find a solution that can be used specifically for a business.

This project will also help to improve my skills in cyber security, secure coding principles and web development which is an area of information technology for me that can be improved upon.

9 Technical Approach

The main tasks of the application is the creation of a secure web application with emphasis on security due to the specialization chosen. Technical approach will include the following steps:

- Requirement gathering.
- User interface design.
- Research of ASP.Net Core security features.
- Implementation and testing of multi factor authentication.
- Using a library to implement QR code generation for multi factor authentication.
- Proper authentication and authorization setup.
- Protections against top vulnerabilities outlined in OWASP Top 10.

- Creation of models, views and controllers to manage data in the application.
- Connection to the database to manage all data including database entries for data management and user accounts.

9.1 Security

9.1.1 Injection:

- Encoding user input before it is passed to the interpreter.
- White list input validation
- Database privilege limits

9.1.2 Broken authentication:

Follow best practices for authentication:

- Using e-mail as username/ID.
- Multi factor authentication
- Strong password only

9.1.3 Broken access control:

- Deny by default
- Logging failed/unusual attempts (IP from a different countr)

9.1.4 Cross-site Scripting (XSS)

- Use of established frameworks that are created to specifically protect against cross site scripting.
- Escaping
- Good content security policy to reduce the chance of a successful attack.

9.1.5 Sensitive data exposure/logging

- Log availability. It is important to keep good logs in order to investigate security failures.
- Monitoring of any suspicious activites such as unusual IP addresses and ensure that threats can be detected quickly. It has been shown that some security breaches can be open for months or years before being exploited or detected.
- Disaster recovery. It is best practice to have a disaster recovery plan in cases such as ransomware or any other cyber attack which may compromise an organization.

9.1.6 Follow best security practices for securing the database:

- Encryption of data.
- Manage database access.
- Monitor database activity.

9.2 Research:

Will be using my obtained skills in C# and performing further research on C#. I am hoping of using C# ASP.NET Core in Visual Studio IDE to develop the application.
I am also hoping to use my skills obtained in “Security Principles” and “Secure application programming” and apply them in my application to ensure that it meets the latest security requirements and provides sufficient functionality to make the application secure such as multi factor authentication.

9.3 Management:

Project will be managed by settings weekly goals in excel.

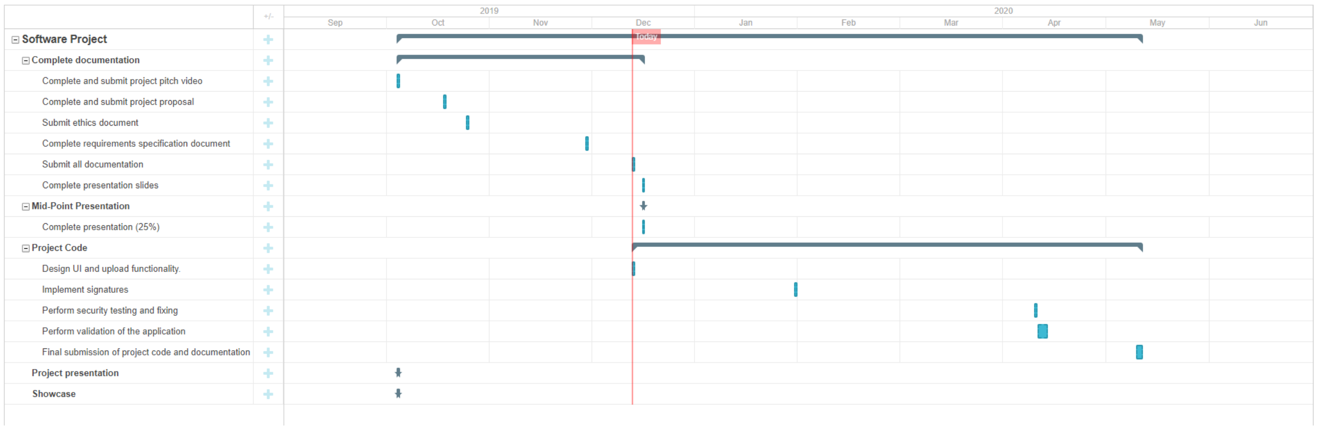
10 Special resources required

Software:

Visual Studio IDE, ASP.NET Core framework, Excel, SQL database will be used.

11 Project Plan

Gantt chart using Microsoft Project with details on implementation steps and timelines



12 Technical Details

Implementation language and principal libraries
C#
ASP.NET Core
MVC
JavaScript
HTML
CSS
Visual Studio IDE
Excel
Microsoft Authenticator/Google authenticator

13 Evaluation

13.1 Testing

The system will be tested in various ways to address the security issues.

User interface testing:

The application is tested to ensure that all the features that are required can be accessed easily from within the web application. This type of testing involves checking if all the fields are displayed, the buttons work and changes are saved correctly and are visible to the user. It checks for any missing features or display errors using different devices such as mobile phone and a tablet to ensure that the user interface is correctly displayed on all the devices and same tasks can be performed as when using a personal computer.

Unit testing:

Individual components of the system will be tested using unit testing methods which will show areas that need to be redone in order to function properly.

Unit testing will be implemented using ASP.Net Core using ModelState validation and ActionResult methods.

Integration testing:

Integration tests will be performed to verify that the components of the application run without errors when dealing with external resources such as databases. Integration testing takes modules from unit testing and applies testing methods to the combinations of the unit testing resources. It checks the combinations of different parts of the application and their interaction.

Functional testing:

Functional testing involves the testing of micro services. This is done to verify that the created application works as it should from the user's side.