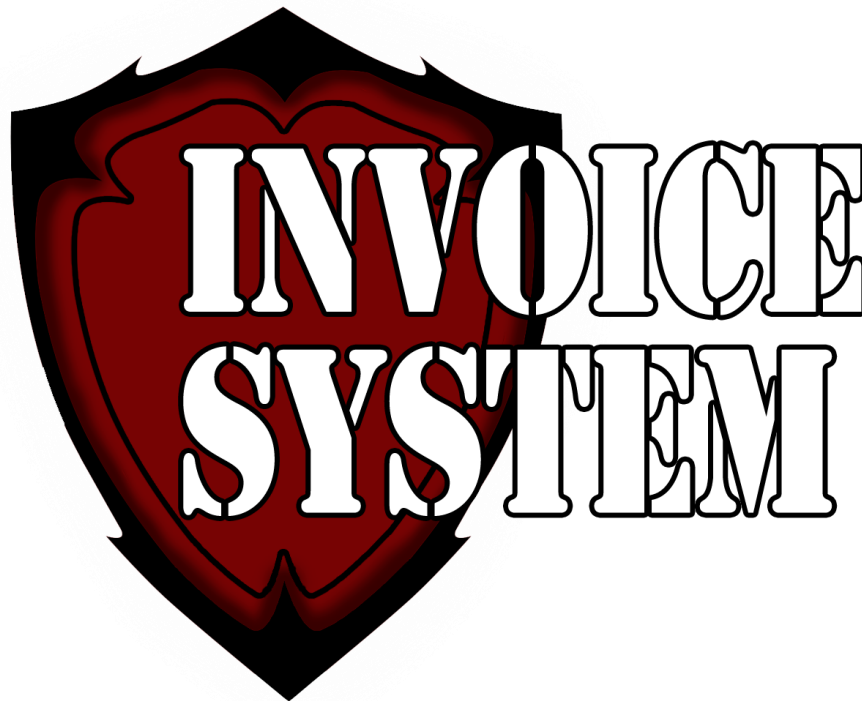


INVOICE SYSTEM

STEPHEN LUKE HARRIS
X13507947



National
College *of*
Ireland

Document Control

Revision History

Date	Version	Scope of Activity	Prepared	Reviewed	Approved
3/10/2016	1	Create	SH		
1/12/2016	2	Update Document	SH		
25/4/2017	3	Update Document	SH		

Distribution List

Name	Title	Version
Sara Kadry	Lecturer	1

Related Documents

Title	Comments
Title of Use Case Model	
Title of Use Case Description	

Table of Contents

Introduction	4
Background	4
Purpose of Document	4
Project Scope	5
Definitions, Acronyms, and Abbreviations	5
Technologies	6
Maven Http Requests.....	6
MySQL Database	6
Apache Tomcat v8.....	6
AES Encryption	6
Eclipse Dynamic Web Project.....	6
Structure	7
System	7
Requirements	8
Functional Requirements	8
Login	8
Create an Invoice.....	10
Stock Management	12
Create Users	14
Add Credit Card	16
View Invoices.....	18
Security Requirements	20
String Validation	20
SQL Injection Prevention	21
XSS Prevention	21
Access Control	22
Session Management	22
Secure Coding.....	23
Third Party Validation (Sort of)	23
Cryptography.....	23
Error Handling	24
Password Management	24
Database Security.....	25
User, Usability and Environmental Requirements	25
Performance/Response time requirement.....	25
Availability requirement.....	25
Extensibility requirement.....	25

Design	26
Visual	26
Database Design	27
System Design	29
Implementation	30
Graphical User Interface Implementation	36
Testing	39
results.....	39
Evaluation	44
Conclusions	44
Future Prospects	45
Appendix	45
References & Other Materials Used	45
Project Proposal	46
Evaluation	50
Reflective Journals.....	51

Introduction

Background

The project initially started following an internship in Thriftify where I worked on both the server and the client side of the website. While working there, I discovered XSS and some SQL injection that could affect the websites confidentiality and some elements of integrity on the sections I made myself. This is when I started implementing server side validation to the project without any prior knowledge of security methods that could be taken.

After specialising in security in my final year of college, I have learned more techniques that further protect the user. The problem with a lot of websites today is they do not contain security elements that protect against some of the most common hacks outlined in OWASP Top 10 Web Application risks 2013-2017. SQL Injection is still a major issue with some web pages that have less funding to set up security measures. An example of this still available is a bus company's website that offers lifts to the airport staff at hours that Dublin Bus do not operate. If you navigate to the 'Find my bus' section on the website and enter any character that my return an error in SQL, the website informs the user of an SQL error. I didn't test any further because I do not want to be responsible for a website hacking, but it shows two examples of OWASP top 10 that the company do not protect against on a regularly used website, Error handling and SQL injection. This is only one example of what malicious activity can happen online. some of the biggest multinational companies have been claimed a victim to Information exposure, Microsoft's Xbox accounts have been hacked in the past which even my own account data has been leaked, Paddy Power, Drop Box... the list goes on. If some of the security features present on my application were present on these huge systems, the account information would not have been leaked.

On the other side of things security can slow the website down and the website needs to be available to users and accessed in a quick and timely fashion. Too much security can lead to slow response times in order to protect the Confidentiality of the user's information. For an example of this, AES encryption can take roughly 2-3 seconds to Encrypt/Decrypt any given value. If we decided that all information located on our server is highly sensitive, we could use our encryption technique to encrypt all of the information. Although this can seem like a good idea to any inexperienced user, this would essentially stack the Encryption/Decryption times to 15-30 seconds. When making an application to be used in the general public, this is not acceptable. The general standard for wait times is 1 second for navigating a website between pages. So for this project I aim to meet this while still using the AES technique. Of course in areas such as logging in, the user will experience a longer delay of about 2-3 seconds due to the encryption.

Purpose of Document

The Purpose of this document is to inform the user of the different aspects of the project that I have created. when looking at the project from an end-user's point of view it may seem that there are not huge amounts of work put into the project when in reality there are multiple different security measures taken when creating the project to insure that the user is safe from any attackers trying to steal the user's information

The Project in short is a Dynamic Web Project which will accept a user login to allow them access to a sensitive data stored on the website's Database. Sensitive data includes credit card information, passwords, etc.

The intended customers are the front end users of the application. The security should be invisible most of the time to users until the software asks for verification from the user in the form of passwords or other methods.

The overall aim of this project is to make a watertight project against possible incoming attacks to protect the user's

information while also making it accessible to users without ridiculous wait times. Due to some time restraints, I'll be using bootstrap to design the website as it is not my main focus to fill pages with eye catching designs but also not leave the website in such a state that it is unusable.

Project Scope

The Scope of this project is to maintain system confidentiality of a Dynamic web project. The system confidentiality is the private data which is stored on the database for each user or underneath a users company. This will be done by preventing users the ability to access files they do not have permission to view using multiple security features which will be given in more detail in the rest of the document.

I plan to protect the main three fundamental security properties of any security system using an array of methods. The three are called Confidentiality, Integrity and Availability or (C.I.A. for short). If all three properties are implemented correctly into the entire system, it will become pointless for hackers to attempt a hack as the wait times to decrypt information can be too long for the reward that will be received for hacking the information (which can take up to something ridiculous as 1,000 years or even more).

Confidentiality means all the data stored on the system that is not publically available, (credit card info is secure, the contact us page of a website is publically available) is secure and not available to any unauthorised user. Integrity follows on from confidentiality with the idea of maintaining the reliability of the information being viewed or in other words, Ensuring the user is looking at the correct information and that it has not been altered to show inaccurate information. Finally, the last being Availability ensures that users that are intended to view the required information are able to in a timely fashion.

Definitions, Acronyms, and Abbreviations

Eclipse: Programming Software used to create the Project

MySQL Workbench: Database software to create and manage databases

Dynamic Web Project: Is a website project that runs on Eclipse. it uses java, and web based programming languages, to work. it sends data back and forth to the website whenever an action is made by the user.

AES: Is an encryption technique that is very secure and a standard.

Database: is a place where information is stored. such details as username, password or any other details stored.

System Confidentiality: The model of only allowing authorised users to view sensitive data on a system. No un-authorised users can have access to the system.

System Integrity: The sensitive data being displayed is factual and not changed either in transport of the data from the server to the user. visa versa.

System Availability: Users who have access can access the data in a timely fashion with a usual time limit of about 1-minute maximum.

Sensitive Data: Data that cannot be shared to the public. (E.g. Credit card information)

Technologies

For this project I'll be using an array of different technologies. From Apache tomcat servers to SQL and others. below are the titles of the technologies used with an explanation of what each one does.

MAVEN HTTP REQUESTS

Http Requests are URL's like what you would see in a web browser that contain information that have information at the end that the server can pick up and process, and if required, can send back some information in XML, JSON or plain text. For this project, HTTP requests are used to verify credit card details through what is supposed to mimic a third party credit card validation system.

MYSQL DATABASE

The Database is managed using MySQL Workbench. The database uses different user profiles as a form of access control. this prevents users from accessing information they should not have access to at any time. Some of the profiles only allow access to the users table with a single select statement. this is a first step to prevent any unnecessary data being leaked.

APACHE TOMCAT V8

the latest version of Apache at the time of writing this is being used.

AES ENCRYPTION

Another Highlight of the project is the high level of encryption that I'm using for the project. AES is a 256-bit rate encryption method that uses both a vector and a key to encrypt and decrypt values. It is resistant against Brute force attacks due to the long computing time. In other circumstances, Brute force would be able to try thousands of passwords per minute, some methods like hashing can be decrypted what seems to be instantly by a program widely available called "jack the ripper". With AES the program would have to wait 2-3 seconds between each attempt. this limits the amount of attempts available per minute to around 20~30. this increases the wait time of brute force attack to years making it not worth an attacker's time.

ECLIPSE DYNAMIC WEB PROJECT

The project itself was built using Eclipse Dynamic web project with Maven functionality added in. Although the Maven functionality is fully utilized in its current state. it leaves open the possibility of creating a service that will connect to an external application for 2-step verification or any other functionality that might be implemented.

Structure

The Document will cover all the different functions that the system can preform on both the front end and the backend. After describing the entire system and outlining all the features I will then cover the implantation of the project and show some screenshots of code that helped create the system. We will then go into testing and some future prospects of the project.

In the final appendix monthly journals are available to view and the project proposal. From reading these we can see how the project has evolved over the course of the year starting out as an extension to a charity companies already live website, to now its own functional invoice system. As I am very familiar with the system present on the Thriftify website, I used the same programs to create my own project while taking some knowledge learned from the system and implementing it to my own.

System

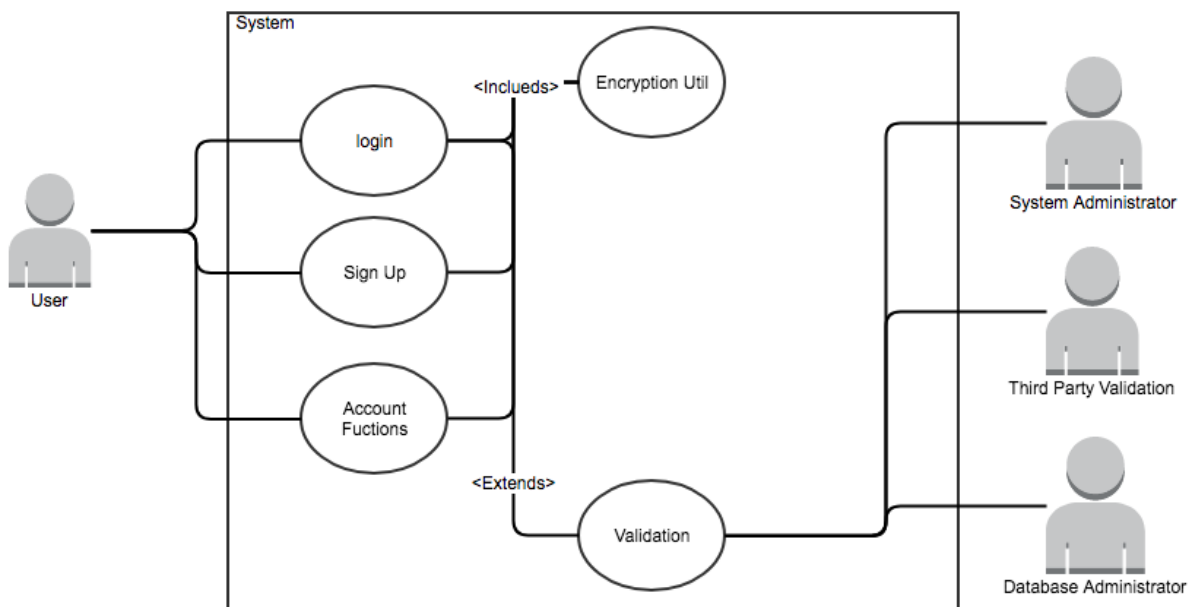


Figure 1

For this section we will concentrate on a few elements in the project related to the system itself. We will cover the functional requirements, security requirements, design and architecture of the system itself.

Above is an image of the entire system design and how each section interacts with each other. In the next following section ill be going through each part in further detail on each function. The Account functions breaks down further into smaller pieces but for the interest of simplicity and space I've combined the functions. Another example of combining functions is the Encryption Utility which also contains the PDF Encryption.

Requirements

Functional Requirements

Functional requirements describe what the system should do. so for this section I'll be putting together a list of functional requirements that I have created in the project that the end level user will be able to see and interact with.

LOGIN

Probably one of the more obvious requirements for a website, but is still needed. the website must accept a username and password when attempting to access their account. more details of how this is secure will be covered in the Security requirements section of the document. This requirement allows the user to access the rest of the functional requirements.

Use Case

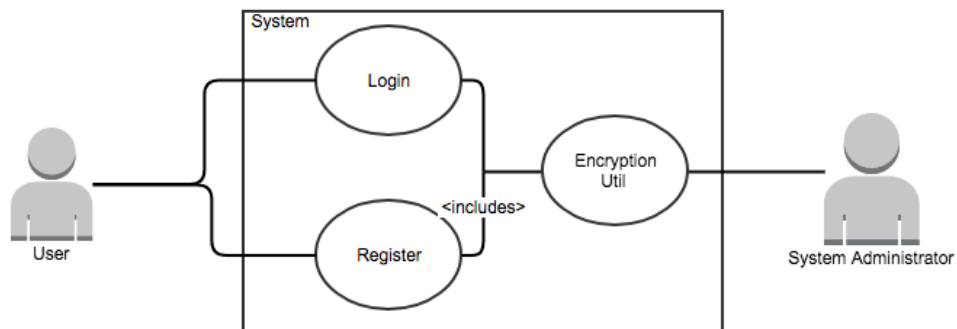


Figure 2

Flow Description

Precondition

The user must already have an account setup to begin logging in with a username and password. the user must also have internet access with a web-browser

Activation

This use case starts when an actor selects the Sign in button on the website.

Main flow

1. The User Selects the “Login” Button on the Home Page
2. The System Prompts the User to to enter Username/Password
3. User Enters Username/Password. (E1) (A1)
4. System Checks Username and Password
5. System Directs User to MainPage.jsp

Exceptional flow

1. The Actor Submits Username and Password
2. The System rejects the user for entering either wrong password or username.
3. If the user persists to enter the wrong password, their account can be disabled.

Alternative flow

1. The User Enters a wrong Username/Password
2. The system Prompts the user to re-enter Username or Password
3. The Actor re-enters information correctly
4. The Use Case continues to next step where left off

Termination

Blocking Accounts after multiple attempts can be done manually by the system administrator. A Log of failed attempts is recorded and what account they are trying to log in to. If it seems they are trying to brute force any given account, the account can be deactivated.

Post condition

The system grants access to the functions they have access to on the main page.

CREATE AN INVOICE

Probably one of the most straightforward requirements, the ability to create and store invoices that will be available to the user upon request. In its current state, the Web Application takes in values from the user and displays them across a pdf that was designed by myself (I'm unfortunately no Picasso). Information from the user's account is taken as well to speed up the process of the form. For example, parameters like username, Company Address, company name etc. are already values that have been entered by the user and do not need to be re-entered again. another major feature about creating an invoice is the mandatory password that is required when creating an invoice. this protects the Confidentiality of the invoice and since this only unlocks the ability to read the invoice it helps secure the integrity of the information stored on the invoice as well. Below is an example of the PDF the invoice will be printed out on.

Company Name

From: admin
24 swords manor court

Invoice For:
Stephen Harris
24 Swords Manor court

Invoice ID: a1DocTitleDocID

Issue Date: 25/04/2017

Swords
Ireland
K24 Was4

PD Number: 12121

Due Date: 02/05/2017

Subject:
Test

Description	Quantity	Unit Price	Amount
Velvet Shirt	2	23.0	46.0
Jeans	3	15.0	45.0
Chocolate Bar	2	2.0	4.0

Subtotal: 96.0

Discount: 0.0

Amount Due: 96.0




Figure 3

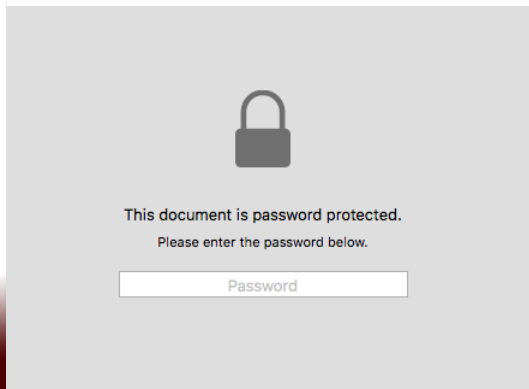


Figure 4

In this current version, it is only possible to have 6 items max per invoice. This will be updated in a future iteration to be able to have multiple pages per invoice. To add stock to the invoice, the user must manage their stock in another section which allows the user to create, update and delete stock from their inventory. The stock will then be available to view when adding the items to the invoice and a quantity chosen. For future iterations this can be changed to display a large amount of items efficiently and easily for the user to manage. but in its current state it shows the items in a drop down menu which will cause issues when looking at lists that may contain 20+ items. The PDF itself has also been updated significantly to now advertise the origin of the invoices website so in future when the application is live it can be clear to users how who made the invoice as a form of advertisement. Other future iterations may also include the ability to upload a personalised PDF so that the customer can have a cleaner looking PDF than the one that is present.

Use Case

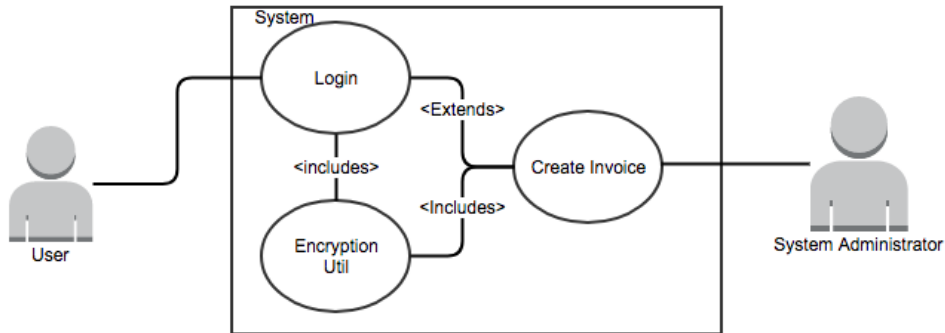


Figure 5

Flow Description

Precondition

The User must be first logged into their account and have stock added to their inventory.

Activation

This use case starts when an actor selects the " Create Invoice" button on the website.

Main flow

1. The User Selects the "Create Invoice" Button on the Home Page
2. System Directs User to fill in Form
3. User Enters Data On Form.
4. System Checks All Data
5. System Directs User to MainPage.jsp

Exceptional flow

1. The User Tries to manipulate HTML to edit other information
2. System Detects user does not own item
3. System Ends user session and logs report to server console

Alternative flow

1. The User does not fill out the form correctly
2. The system Prompts the user to re-fill in the form
3. The Actor re-enters information correctly
4. The Use Case continues to next step where left off

Termination

The termination process should be more than a rare occasion with the user only being forced to be logged out by the system if they try to edit items on the site they do not own by editing html items

Post condition

Invoice Is created and the PDF invoice is added to the Database.

STOCK MANAGEMENT

Stock management is as simple as it sounds, the user will have the ability to add new items, delete old items and update stock quantities. This is just a handy way of keeping track of stock that will work alongside the Invoice pdf described previously. No over the top complications in this section for the end-user as it is just simply Stock management.

Use Case

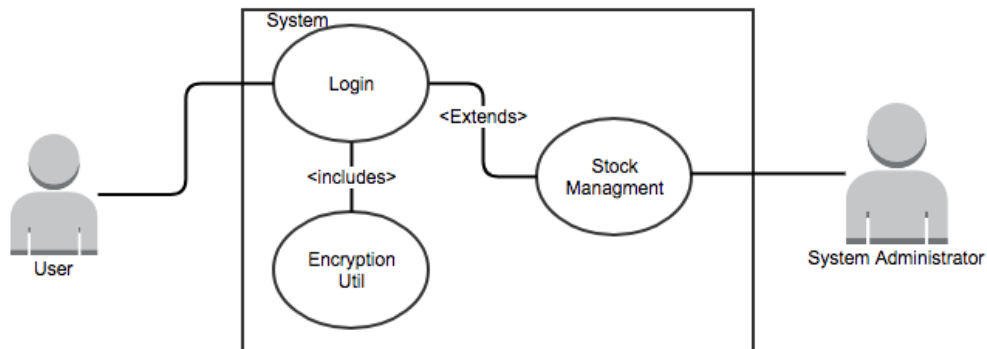


Figure 6

Flow Description

Precondition

The User must be logged in and have access to view the stock file if their account was created for them. to access all the functions, the user must either be given access or created their own company account.

Activation

Begins when the user selects the Stock Management Button located on the main page after logging in

Main flow

1. The User Selects the "Stock Management" Button on the Home Page
2. The System Gets the stock files and re-directs the user to the Stock.jsp page.
3. The User has the Option of entering data to the stock file with CRUD Functionality(a1)
4. User selects the appropriate option for the system to carry out.

Exceptional flow

1. The User Tries to manipulate HTML to edit other information
2. System Detects user does not own item
3. System Ends user session and logs report to server console

Alternative flow

1. The User does not fill out the form correctly
2. The system Prompts the user to re-fill in the form
3. The Actor re-enters information correctly
4. The Use Case continues to next step where left off

Termination

The termination process should be more than a rare occasion with the user only being forced to be logged out by the system if they try to edit items on the site they do not own by editing html items

Post condition

The System updates, Deletes or Adds the stock and the user is presented with an updated page

CREATE USERS

This section can be split into two sections, firstly is the ability to sign up and create a personal account and a new company, the second being adding employees or other user accounts to the system that will work underneath the company name.

In slightly more detail, the signup section is located on the face of the website where the user will login. creating an account is simple enough with the system asking for a company name, address, username, password and some other simple information that will be filled out. if successful, the user is brought to the index page where they can then attempt a login.

For the second version of creating users, the original user how set up the company will be able to add users to the system with another simple layout asking for minimal information about the user. username password, phone number. other information will be taken from the account used to create the user's account like address of the company and what company the user works under. If the user wishes, they will be able to set restraints on the user's account that will stop them from editing stock quantities and creating new users.

Use Case

Sign Up Use Case:

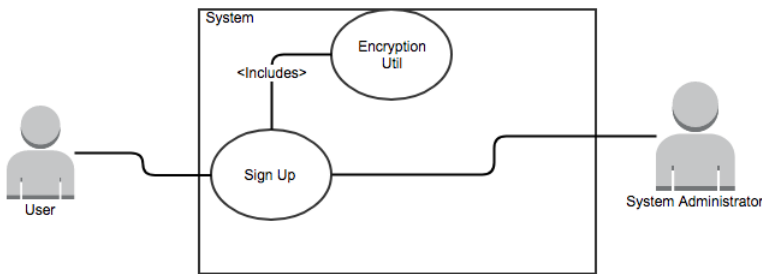


Figure 7

Create User Use Case:

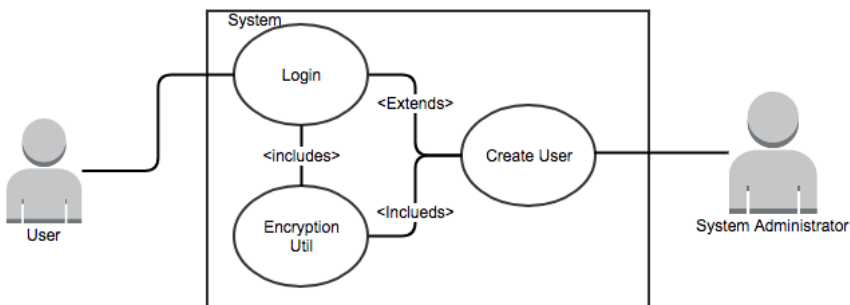


Figure 8

Flow Description Sign Up

Precondition

The User is not required to be signed in. the user needs to click the Sign Up button on the index page.

Activation

Begins when the user selects the Sign Up Button located on the index page

Main flow

1. The User fills in the form presented to them
2. The system checks the details
3. User is redirected to the Index page to log in.

Exceptional flow

1. Only the user can terminate this by selecting the menu option

Alternative Flow

1. The User makes enters some data that does not pass the validation checks
2. The user is prompted to re-enter the information to make and account.

Termination

Termination of this process can only be done by the user if they decide to select the Menu

Post condition

The System creates a user and company based on the information given. This is now a valid user that can log into the system.

Flow Description Create Users

Precondition

The User must be logged in and have access to view the stock file if their account was created for them. to access all the functions, the user must either be given access or created their own company account.

Activation

Begins when the user selects the Create User Button located on the main page after logging in

Main flow

1. The User selects to add a user.
2. System re-directs the user to AddUser.JSP
3. The form is filled out
4. User Selects the Add User Button(a1)

Exceptional flow

1. Only the user can terminate this by selecting the logout or menu option

Alternative Flow

1. The User Selects the Add Restrictions Button
2. User then Gets re-directed to the AddR.jsp Page
3. The User selects what permissions they want to give the user
4. The user selects the Add User Button

Termination

Termination of this process can only be done by the user if they decide to select the Menu or Logout button

Post condition

The System creates a user with the permission's created by the user. This is now a valid user that can log into the system.

ADD CREDIT CARD

in this section, although in the current state of the program the information is not used, the user will be able to add a credit card to the system in order to charge customers for purchases. this section will work in order to store the cards but is currently a stand alone feature that can have future prospects. in a later section I will revise the security created around this section and describe why it was essentially added to the project.

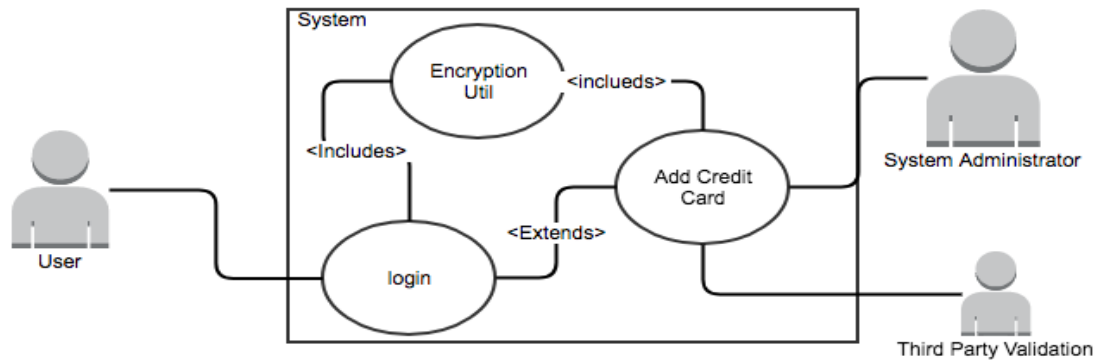


Figure 9

Flow Description

Precondition

The User must be logged into the system and be viewing the main page.

Activation

Begins when the user selects the Add Credit Card Button located on the main page after logging in

Main flow

1. The User selects the Add Credit Card Button
2. The System Directs the User to the Page
3. User is prompted to enter the correct information
4. The System Validates Card Server Side
5. System Sends appropriate data to Third Party Validator
6. System Receives a true or false value
7. System adds or updates the credit card information.
8. System Redirects User to Main Page.

Exceptional flow

1. Can only be done if the user navigates out of the page by the Log out or menu button

Alternative flow

1. Validation notices a fault in the information and prompts user to re-enter info
2. The User Enters the correct info

Termination

Only the user can terminate the process by navigating out of the function

Post condition

System creates/updates the card and user is redirected back to the main user page

VIEW INVOICES

this final functional requirement allows the user to download the PDF that they created and is listed under the user's subject field and date created. The PDF is stored as a blob in the database, but since it is not possible to encrypt a blob without destroying the document itself I came up with a clever way of securing the data so that it can not be retrieved from the database easily. this will be covered in the Database security section in the security requirements as this section only covers the end user requirements. but basically the user will click a button that will allow them to either download the PDF from the website or take them to view the PDF depending on the browser that they use. Each of the PDF documents are encrypted and password protected preventing the data from being leaked.

Use Case

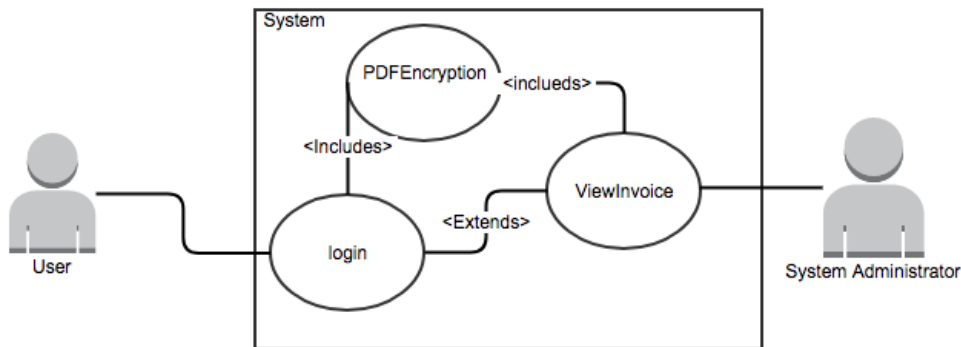


Figure 10

Flow Description

Precondition

The User must first create an invoice which requires them to add stock to their account first.

Activation

Begins when the user selects the View Invoice Button located on the main page after logging in

Main flow

1. The User Selects the “View Invoice” Button on the Home Page
2. The System Gets the stock files and re-directs the user to the ViewInvoice.jsp page.
3. The User selects the Appropriate Invoice Available from the list
4. Depending on the browser the Invoice is downloaded via PDF
5. The User then must enter their password to view the file.

Exceptional flow

1. If the User does not have the password, the user cannot open the PDF

Alternative flow

1. If the user is on a old browser, they will be re-directed to the PDF Document in the browser.

Termination

Termination is yet again up to the user via the navigation buttons

Post condition

The user can either download another pdf or choose to navigate out of the page

Security Requirements

in this section similar to the previous section, I'll be going over the security requirements for the project. This section we will take an in-depth look at security functions that are implemented throughout the project. Some functions are located in multiple areas of the project (e.g. String Validation), so for the purpose of simplicity, I will be condensing these together so I don't repeat myself and can explain what each security feature does that will be simple to understand for anyone who is not a familiar with security techniques.

STRING VALIDATION

```
private boolean usernameTest(String test) {
    if (test.matches("^[a-zA-Z0-9_\\.]*$")) {
        return true;
    } else {
        return false;
    }
}

private boolean passwordTest(String test) {
    if (test.matches("^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])[a-zA-Z0-9]{8,}$")) {
        return true;
    } else {
        return false;
    }
}

private boolean emailTest(String test) {
    if (test.matches(
        "(?:[?:\r\n]?[ \t])*(?:[?:(?:[^\s@;:\.\\"\\\\\ \000-\031]+(?:
    ] else {
        return false;
    }
}

private boolean phoneTest(String test) {
    if (test.matches("^[0-9]{6,20}$")) {
```

Figure 11

One of the simpler security techniques to implement into the project, string validation is a way of taking any user input from a website that will be passed to the server and ensuring that there is no malicious code added to the text that might compromise the integrity of the system. This prevents things like SQL injection from affecting the system. For example, when logging in, or on sign up, it is not allowed to use characters like % in the username. This is however allowed in the password for the simple reason that the password is encrypted which completely changes the output of the string entered.

String validation is located in multiple places across the servers Presentation controller for different values and in some places in the backend where needed. It works in most cases by comparing a value to a regex which will figure out if the string breaks any rules set by the regex. A classic example of this is in the password where the user must enter a password with a lower and uppercase letter and a number. also it needs to be 8 characters or longer with a limit of 25 characters. It is probably the simplest form of ensuring protection against XSS (Cross Site Scripting, Another String Manipulation Attack) and SQL Injection.

SQL INJECTION PREVENTION

This is an extension of the last point; I have used some trivial techniques in preventing SQL injection. SQL injection is the process of entering SQL commands in the text boxes of a website that would reveal information the user does not have access to that would affect the Confidentiality of the website.

One of the techniques I use is simply comparing the user's input against the output of the Database when attempting a login. this works because if the user somehow manages to SQL inject into the sign in (which should be impossible anyways as the username and password must match and the password being encrypted before comparing) the result of the the database output must then equal the input of the user. For example, for a trivial SQL injection technique entering %' or '1'='1 into the username bar will not return true as let's assume it manages to find a user of the name "Michael_J_Fox" for arguments sake. the system will then check to see if the username "Michael_J_Fox" equals "%' or '1'='1", which of course isn't true making the login attempt invalid and informing the user of an invalid login and a message displayed on the server log of the user's IP informing us of the attempted failed log in.

Another method used is the Database security itself. I have created multiple user profiles in the system that only allow the user to access certain information when entering data into the text boxes. this is a form of access control. For example, on the login page, the user connects to the database as a user who only has access to perform a select statement on the user table. this prevents the user from using other commands like unions or insert statements that may affect the integrity of the system.

XSS PREVENTION

this is a way of manipulation the script tags in a HTML website to return a value to the attacker that will give them more information than intended. There are multiple methods used to prevent this attack that will be covered in greater detail in later sections of this.

```
<%  
int timeout = session.getMaxInactiveInterval();  
response.setHeader("Refresh", timeout + "; URL = index.jsp");  
%>
```

Figure 12

Session management helps combat against this as one of the outputs of a typical XSS attack is hijacking a user's session cookie to give the attacker the ability to access the session. above is an image showing the implemented feature. so for this I have set a time of 4 minutes to be left inactive for before ending the user's session. also when logging out, the user's session will be terminated and a new session cookie will be assigned on the following log in. this can give the attacker a very short window to attack another user if they manage to achieve a XSS attack.

Also we do not allow the use of the main factor that causes XSS which is the following two symbols "<>" when creating any user accounts. another factor is not allowing users to see other user's information regardless of what company they belong to. this prevents users from attacking other users on the site and stops XSS in its tracks. The only place you'll notice that will allow special characters is when creating a PDF document as at no point during this phase is any of the data used in the Database or in the Browser meaning it is only printed on the pdf and stored until the pdf is downloaded again.

ACCESS CONTROL

Access control is present in two areas of the project. Firstly, it is implemented in the area of the database described previously. Essentially this is just the creation of profiles that can login/Connect to the database. For example, the Login Account will only have access to the user table with a single select statement or the stock management will only have access to the stock file with create, read, update and delete functionality. this prevents any unnecessary leakage of data across the database if someone does manage to retrieve the information, there will be minimal damage.

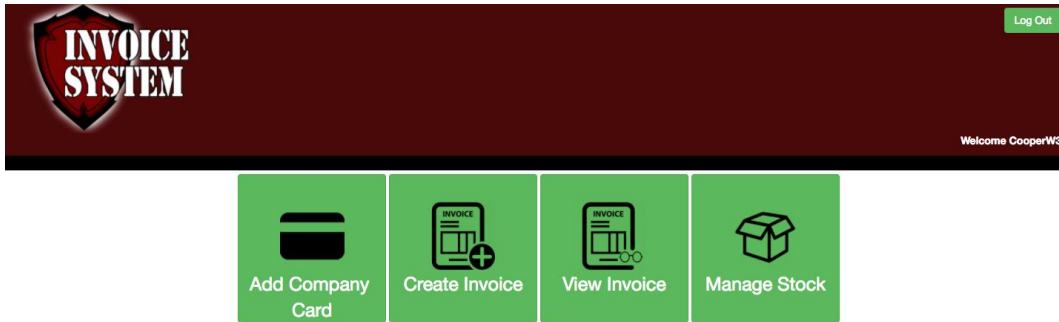


Figure 13

Second instance of this is on the users themselves. some users will be linked to a second user table which will describe each user's permissions if they have any restrictions on their account added by the company creator on the website. Some of these include selling/adding stock, ability to create new stock, delete stock etc. if these values are undefined for the user it is assumed they will share the same functionality as the original creator of the company on the site and will be able to access the same functions like creating new users. As we can see from figure 13 above, this user has no access to create a new user.

another example of this is not allowing users to access pages they do not have access to. this ties in with session management, if a user decides to manipulate the URL of the page to try gain access to site functionality they do not have access to (e.g. typing in AddUser.jsp to the end of a Url in Figure 13s case), the website will recognise this and restrict the user from accessing the page and redirecting them to the index page while also ending the user's session. Ending the user's session is just a way of annoying the attempt of string manipulation preventing the user from attempting the attack multiple times in quick succession and hopefully deterring them from trying again.

SESSION MANAGEMENT

Session Management is applied across the the website as a whole. If a user is not in a current session that they have logged into they will not have access the majority of the pages an active user will have. public pages will not have this restriction.

Also a time-out function will be in place for each session (Figure 13). If a user is inactive for a long period of time the user's session will end and will be redirected to the index page where they will need to log in again and get a new session cookie.

Update Post Testing:

After some testing done with the users, the session timeout was set to 4:00. this was due to some confusion by the users at each given section when first accessing the site. each time the user clicks a button on the site, the timer is reset.

SECURE CODING

```
private static String initVector = "&JtZ7/Zy{<-C2L^X>";

private static String decrypt(byte[] key, String encrypted) {
    try {
        IvParameterSpec iv = new IvParameterSpec(initVector.getBytes("UTF-8"));
        SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");

        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec, iv);

        byte[] original = cipher.doFinal(Base64.decodeBase64(encrypted));

        return new String(original);
    } catch (Exception ex) {
        ex.printStackTrace();
    }

    return null;
}
```

Figure 14

Simply put, secure is regarded as restricting access to certain functions within the Web Application to prevent any users from gaining information that is deemed sensitive. Using protected, private and public methods in the correct area's can prevent attackers from accessing the important information in certain areas of the server. As we can see from the above image, secure coding techniques have been implemented into the Encryption elements of the application, other areas such as user info stored on the session and items lists are also kept private and can only be pulled out from internal classes keeping the information safe and keeping confidentiality.

THIRD PARTY VALIDATION (SORT OF)

Third party validation can be loosely used as a security requirement as it is almost untrue for this project. what I mean by this is that I have also created a second project that can send and receive HTTP requests that will communicate to my project to verify credit card details. This is supposed to mimic the use of an actual 3rd party software that might be used to validate credit cards like what AIB provide. The function simply checks if the credit card 16 digits long and sends back a true or false string accordingly. In a real world situation, the bank would check if the number is part of a valid credit/debit card and return the appropriate value to inform us if the card is valid or not. this function is more in place to show that I am aware of the third party validation and the project will implement the function in a future iteration of the project, but for this prototype it was an additional expense that was not nessasary.

CRYPTOGRAPHY

```
IvParameterSpec iv = new IvParameterSpec(initVector.getBytes("UTF-8"));
SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");

Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
cipher.init(Cipher.ENCRYPT_MODE, skeySpec, iv);

byte[] encrypted = cipher.doFinal(value.getBytes());
String Final = Base64.encodeBase64String(encrypted);
```

Figure 15

This is what I would consider the holy grail of the project. within the project I have implemented AES encryption which is a very high standard encryption technique that is nearly impossible to crack. This is due to the long time spent between

attempts compared to other methods. In regular situations where Hashing is used, a brute force attack can be used to crack a password encryption key. this is done by guessing the password thousands of times per minute. programs like "John the ripper" can crack a password hash it what seems to be instantly. But in the case of AES encryption, each attempt can take up to 2-4 seconds per guess. this shortens the guesses per minute from thousands to below 30 per minute. and with the added bonus of making the user make strong passwords, there are millions of different possibilities a password may be. the wait time of cracking a password or any sensitive information encrypted now escalates to hundreds of years at a realistic minimum. this wait time is really not worth the reward of the information gained, so this rules out a successful brute force attack and insures Confidentiality of information with a small sacrifice of 2-4 seconds of availability.

The information that I Encrypt includes all sensitive data in the database. Passwords, Phone numbers, Credit Card Info and so on. This protects all the user's information even in the event of a breach which should be nearly impossible in the website's current state.

The final piece of information I encrypt is the PDF document itself. the PDF is encrypted by the creator of the document before submitting the data to the database by entering a mandatory password at the bottom of the form. When downloading the document, the Encryption is applied to the document protecting the Confidentiality of the document so that nobody without a password can decrypt and/or view the document. The Invoice itself contains sensitive data that could be detrimental if it were to leak to a possible rival company or to the general public. for this reason, the password is applied to the document. Again, AES is near impossible to decrypt with today's technology as long as the password is a lengthy strong password which we enforce on the site on the server. if a weak password is submitted, the user will have to refill the document out and re-submit it with a stronger password.

As a note, every piece of information on the Database is encrypted by the user's password from their account with the exception of the company ID in the PDF table. This makes it possible to protect the user's information even if one user does so happen to be breached. This protects each and every user with little to no cost of time to the user itself.

ERROR HANDLING

Error handling is probably one of the most prominent features in this application. the application itself has countless occurrences of error handling. This is an issue as much as it is a great feature. this is because it can be next to impossible to tell if 100% of all errors are being handled in a secure manner. the only real way of telling they have all been found would be countless hours of testing to ensure that in the event of any failure, the Web Application will be able to deal with the situation. When examining the code, you'll notice that some features even rely on an error happening to continue working for the end user. Examples are areas where the user is not required to enter information into the site like in the invoice. when no information is given on some fields, and error is caught and the value is set to null and is later dismissed by the program when filling in the PDF document.

PASSWORD MANAGEMENT

A short title as it is heavily tied into String Validation, Passwords across the entire site whether you're creating a new account or encrypting a Pdf document must be 8 characters or longer and contain 1 number, 1 uppercase letter and 1 lowercase letter. This is to ensure the integrity and confidentiality of the company's information so that no information is leaked.

DATABASE SECURITY

The Database Security was another focal point of the project. I wanted to ensure that all the users data was secure and even in the event of a breach that there was still another layer of defence that would stop useful information from leaking. For the last line of defence, I against the breach I have in place the AES encryption on all sensitive data like passwords or credit card numbers as stated in the Cryptography section above. As another line of defence I have in place another factor of Access Control. there are multiple user accounts used in the Database that only have access to the specific tables needed when accessing the database depending on the function you are accessing. so for example the Login Function only has access to the user's tables and to perform a select statement ruling out and "union" tags that might give the attacker any other useful information.

User, Usability and Environmental Requirements

PERFORMANCE/RESPONSE TIME REQUIREMENT

The User should be able to access their account in a quick and timely fashion. The time taken to login after pressing submit should take no longer than 3-4 seconds maximum.

AVAILABILITY REQUIREMENT

The User should be able to access their log in assuming there is an internet connection available. the data being sent between the system and the client isn't large in size so it should respond in a timely fashion. The Encryption and Decryption of data sent between the client and server should be almost seamless to the end user.

EXTENSIBILITY REQUIREMENT

Encryption Methods are always updated. keeping up to date on encryption methods will allow myself to continuously keep the information safe from attacks. as well as this, new methods of verification may arise which can be implemented into my project at a future date. Encryption is a way to protect the data being sent between two systems over a network or even within a network.

Design

VISUAL

Although the design can be an important role in creating a user interface, for the purpose of this project I have not been focusing on the design greatly. throughout the project you'll notice blank spaces and pages that look unfinished. I'd even argue that the logo itself is a work in progress and may come across as cheap. But a general theme is still present in the design. The general overall colour scheme of the site is a dark red, black and white. Logo itself although done for the prototype stage can easily be re worked in the future if needed as it has done already in previous iterations as seen by the two images on the right hand side of the page (Figure 16 the new logo and Figure 17 the old one)

After some constructive criticism about the sort of website I was making and how the logo represented that website, I'm glad now that I went back to the drawing board to re-work the logo in Photoshop and change the entire look of the website. The website before may have been mistaken to be an adult website with the colour scheme and font style which is not the sort of message you want to bring across to customers.

Web pages follow a similar basic layout amongst the pages with a top bar used to to show the logo and some site navigation buttons that allow the user to logout or login and navigate back to the main page once logged into the website. below is an image of the index page which the rest of the pages will follow design wise.

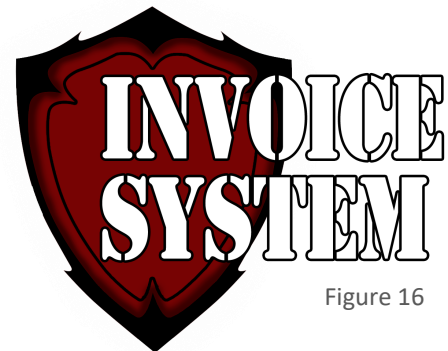


Figure 16



Figure 17



Welcome to the website!

If you are new to the site, check the Sign Up button to create an account ☺

Figure 18

DATABASE DESIGN

For the Database itself there is a simple yet effective layout. Below is an image of the design of the database itself and I'll go into further detail what each section entails.

As we can see from the image to our right, the database contains an array of different tables some of which are actually redundant. The tables we will be focusing on for this section are the Credit Cards, PDFDocs, userRestrict, Items, users, userRestrict.

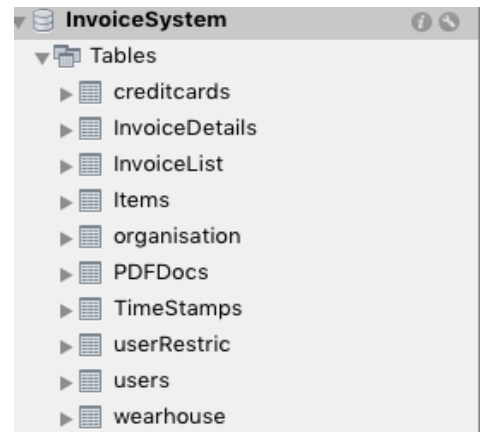


Figure 19

The rest can be considered either redundant due to design chance or do not contain any overly valuable information that needs protecting

Firstly, the user table and the userRestrict Table. these two tables contain information related to the user's account that is vital when using the website. as we can see from the screenshot below there is important information in the user's account like the obvious username and password as well as some others.

idusers	username	password	email	phone	address	active	compld	DateMade	attempt
1	admin	UNP42UGOegr/DkCzPQ8ucA==	admin@admin.com	l=wc9UDQ6me+VCpkObFQD=fC	24 swords manor court	1	1	2016-01-02	15
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 20

as we can see from this screenshot there is not a whole lot of information available to view in the event of an attack. Passwords and mobile numbers are both encrypted to prevent any personal information being leaked about the user. Other information that is used currently is the Username, Address, Compld and active. Compld acts as foreign key although not set up as one in the database, What I mean by this is that the tables are linked through java code and not declaring forging keys in the database itself. this key links user to specific companies so that they can update stock and users under the specific company name. The other data like active is a way of disabling an account so that the user can no longer log into the system. the reason for not just deleting the user is so that any previous reports of which user created what stock, invoices etc. is not lost because the user ID no longer exists.

The final few headings not mentioned from the user table are in place for future iterations of the project. "attempt" will give a max amount of attempts a user can try to log into a specific username before the account is locked. the account can then be unlocked through the email account. this is not currently active in the current build of the site.

the next section is the userRestrict which is short for user restrictions. this enables the creator of a company on the website to set up users with less access than himself with the ability to restrict the user around creating a new user or allowing the user to create, read, update or delete stock from the website.

id	userID	stockC	stockR	stockU	stockD	userC
	NULL	NULL	NULL	NULL	NULL	NULL

Figure 21

we can see that currently for this there is no users added as there is no users with restrictions as of yet. but essentially what will happen is when the user logs into the website the web server will check to see if the user's id is located in this

table, and if true will apply these rules to the user's session. there is validation at each section of the process of each function that it can block so even if the user attempts to manually enter the URL to the specific webpage, it will redirect the user to the Index page immediately. even still if the user manages to make it past this verification element, the web server will deny the access to the database. the different columns from left to right starting from userID are present for the following reasons:

UserID: A foreign key to link the table to the Users table.

StockC: A Boolean value that either blocks or allows access to the created user to create stock for the specific company.

StockR: Gives the user the ability to read what stock is available. if the user does not have this access, they cannot read what stock the company holds and can only create Invoices. This will also prevent users from performing any of the other functions like Creating, Updating or Deleting information even if they are given the permission to.

StockU: This gives the user the ability to update the stock. this can be any of the sections that are found in the items tables found in the database like the quantity, name or category. If the user does not have this permission, they can still edit the values on their screen but it will not be made permanent on the database and will not affect any other part of the system, this will literally just be editing what the user can see keeping the database confidentiality.

StockD: Can give the user the ability to delete stock from the site.

UserC: Gives the user the ability to create new users under the company name. they can also assign users with higher permissions than what they have leaving a small vulnerability to the company's account if they wish to stop that user from manipulating stock files as well. But this section should only be checked to users who are highly trusted in the company anyways.

id	companyid	DateAdded	PDF	name	pass
▶ 19	ah4bFMcpzpeVyds1TdiG6g==	06/05/2017	BLOB	Test2	McY17cPJqvJAHUNK4tFzg==
20	htsP9clxO4a/oZGCeWikwA==	06/05/2017	BLOB	Subject	TtV7NcALkv6ficXiugMOeg==
	NULL	NULL	NULL	NULL	NULL

Figure 22

Here we can see the PDFDocs Table that contains the required information about the Invoice. for example, the Name is taken from the user's subject field. it is validated as well to ensure there is no SQL Injection attempts against the system. The BLOB is the PDF itself in Bytes and the Company in which the PDF is stored under is protected by generating a key based on the account ID and a password set by the user on the document itself.

id	category	description	cost	quantity	companyid	Warehouseid	DateMade	userid	active
▶ 1	Clothing	Velvet Shirt	23	21	1	NULL	2017-04-28	1	1
2	Clothing	Jeans	15	34	1	NULL	2017-04-28	1	1
3	Food	Chocolate Bar	2	400	1	NULL	2017-04-28	1	1
4	Electronics	iPad Mini	200.99	12	1	NULL	2017-04-28	1	1
5	Electronics	iPhone 5s	299.99	60	1	NULL	2017-04-29	1	0
6	Electronics	iPhone 7s	299.99	20	1	NULL	2017-04-29	1	1
7	Adventure	Crossbow	150.5	12	1	NULL	2017-05-02	1	1

Figure 23

id	userid	CreditcardNum	expDate	cvv	DateMade	CardName
▶ 1	1	7ZGmzs3KZFjsntzAKAp8hunUdiREuewMlsTy...	bg4/31KY2jx/+bfGzjs2IA==	1swK6KhOp98zVbvCS9Eqzg==	NULL	NULL

Figure 24

the final two images ,figure 23 and 24, are the items and the credit card number data. Items is as simple as it looks. it holds the specific data related to each item with the only thing needing explanation is the active section. this prevents any loss of data when viewing reports about the system. when deleted from the users point of view, it simply changes the Boolean value to false or 0 and the user can no longer view the item to edit the value. this can be seen with the iPhone 5s.

Credit cards is the second image present and contains the values of the credit card data. yet again the data regarded as sensitive is encrypted using AES encryption so prevent any data from being stolen.

SYSTEM DESIGN

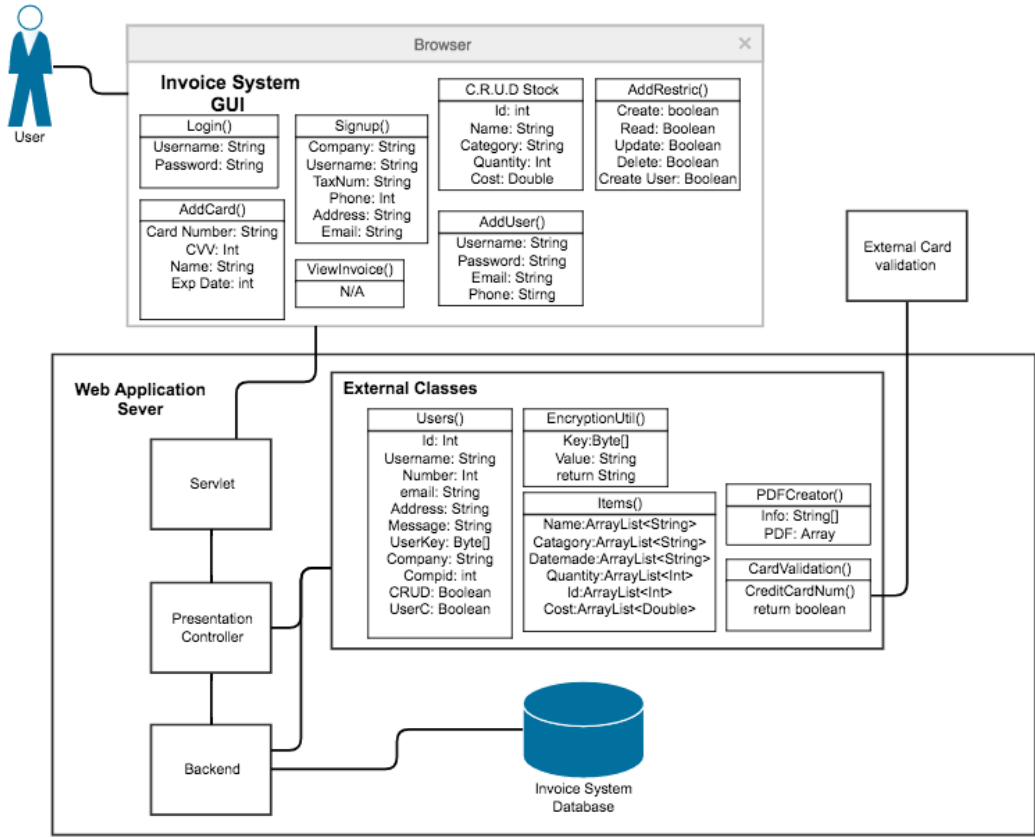


Figure 25

SYSTEM ARCHITECTURE

the Design and architecture of the system above is a look at all the components that exist in the current build of the project. Following is a description of what each section contains and where the computing for each section is located

The user shown in the top left hand corner of the system you can see is accessing the web browser. This actor. The Actor interacts with the browser to perform the functions seen inside the browser field and the information is sent to the servlet. JSP pages work by computing the data on the page itself on the servlet, and sends the result to the user in HTML. the Servlet collects the information through the URL in a form post and then sends the user a new page based off the information given by the form.

All Data on the site follows this general process when going from a form to the database.

User Input → Validation → Computing → Backend → Encryption → Database

and the same in reverse with decryption when retrieving information.

The servlet itself is a core element of error handling, data collecting and some small validation when necessary. the Servlet collects the data from the user ensuring all data needed is present and send it to the Presentation Controller where most validation occurs and any data processing.

The Presentation controller then takes the information from the Servlet and then handles most of the validation. if an error is found the user is redirected back to the page from which they came and are forced to re-enter the information to be re-submitted to the Servlet. If the Input makes it past validation, the presentation controller will then either compute the info and or send it to one of the external classes which either do further commuting, external validation, create PDFs, encrypt etc. when the information is returned to the Presentation controller, the data is then sent to the Backend. The presentation controller also handles data received from the database, it will either use the data to compute info for the user or send it back to the user usually through the ItemList class or the Users class located in the external class.

The Backend is used to mainly deal with information coming to and from the database with most of the encryption happening at this stage as well. As a second level of security in case any users attempt to gain information from other users, all database queries are made secure by comparing the user's ID and Company ID on each query to ensure that the data they receive belongs to them. as this information is taken from their session on the server side, it is impossible to manipulate this information directly. an Attack must hijack another user's session to gain information, which is kept secure by invalidating every session after 4 minutes of inactivity or by the user logging out of the account.

The Database is where the data is stored. all the data is kept in here and is secured by different user accounts that have access to different tables in the database. No encryption of the database takes place in the database itself, all the data in encrypted on the server and then sent to the database to be stored. the encrypted data is then sent back to the server where it is decrypted for the server to use.

Implementation

There is a lot of code and different algorithms present in the project, most of them are already explained in previous sections to prevent myself from re-iterating myself ill be giving a brief description on those algorithms

The first algorithm is the backend login. Although this is usually a simple task to complete I have made some adjustments to make the site more secure for the user.

```

try {
    checkUser.setString(1, username);
    checkUser.setString(2, password);
    rs = checkUser.executeQuery();
    while (rs.next()) {
        // below line prevents sql injection because " %" or
        // '1='1 does not equal what ever the database will
        // return."
        // we dont need to check the password for sql injection
        // because we are comparing incrypted values.
        if (rs.getString("username").equals(username)) {
            if (rs.getBoolean("active") == true) {

                // basic information
                user.setName(rs.getString("username"));
                user.setId(Integer.parseInt(rs.getString("idusers")));
                user.setEmail(rs.getString("email"));
                user.setAddress(rs.getString("address"));
                user.setCompId(rs.getInt("compId"));
            }
        }
    }
}

```

Figure 26

we can see the code itself gives the reader a description of what is happening. But essentially this prevents any SQL injection techniques from being implemented as the input string is compared against the output string of the database. Using methods like this can make it impossible to SQL inject into the database. For the password the string is encrypted before comparing it against the database value. This prevents SQL injection as the entire string has now changed to an encrypted value.

```

// check restrictions
int TMP = 0;
System.out.println("User ID: "+user.getUserId());
try {
    checkR.setInt(1, user.getUserId());
    rq = checkR.executeQuery();
    while (rq.next()) {
        user.setStockC(rq.getBoolean("stockC"));
        user.setStockR(rq.getBoolean("stockR"));
        user.setStockU(rq.getBoolean("stockU"));
        user.setStockD(rq.getBoolean("stockD"));
        user.setUserC(rq.getBoolean("userC"));
        TMP = rq.getInt("userID");
    }
} catch (Exception e) {
    System.out.print(e);
    user.setStockC(true);
    user.setStockR(true);
    user.setStockU(true);
    user.setStockD(true);
    user.setUserC(true);
}

```

Figure 27

In the following screen shot we can see how the users restricted information is gathered. What happens is if the output of the previous database query is true this will then attempt to find if the user has any restrictions applied to the account. If it turns out that there is no database values, the code will catch the error and give the user full access as there is no restrictions present.


```

public void encryptPdf(String src, String dest, String Password) throws IOException, DocumentException {
    PdfReader reader = new PdfReader(src);
    PdfStamper stamper = new PdfStamper(reader, new FileOutputStream(dest));
    stamper.setEncryption(Password.getBytes(), OWNER,
        PdfWriter.ALLOW_PRINTING, PdfWriter.ENCRYPTION_AES_128 | PdfWriter.DO_NOT_ENCRYPT_METADATA);
    stamper.close();
    reader.close();
}
}

```

Figure 28

the image above contains the function that encrypts the PDF document to ensure that the information on the PDF is confidential to the creator of the document. But even still, the password only unlocks the PDF as a user and not an owner. This protects the Integrity of the document a little bit more. Of course the user can still just screen shot the PDF affecting this but it is still an extra layer of security.

While sticking on the subject of encryption the next “EncryptionUtility” is a core feature of the application. The application itself uses this function when ever it needs to encrypt and, when I reluctantly, need to decrypt data I don’t like decrypting data as I feel like it leaves the information open to be stolen if a user’s device is not secure. The only time as of yet the decryption utility is used is when decrypting the password given by the user when creating the PDF document. Below is a screenshot of the encryption utility with some examples of secure coding principles.

```

private static String decrypt(byte[] key, String encrypted) {
    try {

        IvParameterSpec iv = new IvParameterSpec(initVector.getBytes("UTF-8"));
        SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");

        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec, iv);

        byte[] original = cipher.doFinal(Base64.decodeBase64(encrypted));

        return new String(original);
    } catch (Exception ex) {
        ex.printStackTrace();
    }

    return null;
}

private static String encryptThis(String value, boolean internal, byte[] key) {
    try {

        /*
        *Code was used to send full credit card details across api..
        */
        else{
            initVector= "7cpv@9GDN4f#qf&J";
            String tmp = "CreditCardsareco";
            key = tmp.getBytes() ;
            System.out.println(key);
        }
        /*

        IvParameterSpec iv = new IvParameterSpec(initVector.getBytes("UTF-8"));
        SecretKeySpec skeySpec = new SecretKeySpec(key, "AES");

        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec, iv);

        byte[] encrypted = cipher.doFinal(value.getBytes());
        String Final = Base64.encodeBase64String(encrypted);

        return Final;
    }
}

```

Figure 29

I've managed to just about squeeze both encryption and decryption into the screenshot. But we can see for both the user needs to supply a String value for the Utility to encrypt/decrypt and a key that the program will use to encrypt/decrypt the value. What we don't see is a Key Generator that is also in play that creates these keys for the two functions to use. the Key Generator creates a 16-bit Key and combines it with a vector to create 256 AES encryption value.

I don't think it'd be fair if I didn't show off some Validation in the project after talking about it so much In the rest of this. So the screenshot below contains some of the main classes that can be used for different validation across all areas of the site.

```
private boolean usernameTest(String test) {
    if (test.matches("[a-zA-Z0-9_-]*$")) {
        return true;
    } else {
        return false;
    }
}

private boolean passwordTest(String test) {
    if (test.matches("(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])[a-zA-Z0-9]{8,}$")) {
        return true;
    } else {
        return false;
    }
}

private boolean emailTest(String test) {
    if (test.matches(
        "(?:(?:\\r\\n)?[ \\t])*?(?:[^\t@,;:\\\\|\\\".\\\\\\\\] \\\\000-\\\\031)"))
        return true;
    } else {
        return false;
    }
}

private boolean phoneTest(String test) {
    if (test.matches("[0-9]{6,20}$")) {
        return true;
    } else {
        return false;
    }
}

private boolean addressTest(String test) {
    if (test.matches("[A-Za-z0-9 _]*[A-Za-z0-9][A-Za-z0-9 _]*$")) {
        return true;
    } else {
        return false;
    }
}
```

Figure 30

We can see the validation uses a simple regex which checks to see if the input "Test", which is any string I decide to feed the function, matches the text afterwards. The text is known as a regex. In username it checks to see if the string contains letters or numbers. If there any other foreign letters to this, a space or symbol. The function will return false. Password checks to ensure that there is at least one number, uppercase latter and lower case letter. Email as you can see is very long, so long in fact that what you can see from the screen shot I would estimate is not even one tenth of the actual length. This is so long as it accounts for any non Latin letters unlike the rest of my validation. But in short it needs three or more letters followed by a @ symbol, then three or more letters followed by a . and another two letters. The rest of the regex's follow a similar suit and ensure that no characters are going to effect the security of the system.

The final algorithm that ill be showing off is the PDF Creator. This is what creates the layout of the pdf and allows users to enter the information. For this I had to use trial and error in order to get the correct placements of the text to be printed on the pdf. The text is placed on the PDF using X, Y values across the page. In the below screen shot we can see some huge amounts of information being sent to the PDF Creator and how each item is placed on the pdf

```

public void createPDF(String[]info, String ad2, String postcode,double discount,String due,ArrayList<Integer> sItems,ArrayList<Integer> quantities, ItemsList
//Users/sluekeharris/Documents/workspace/InvoiceSystem3/WebContent/PdfTemp
PdfReader reader = new PdfReader("/Users/sluekeharris/Documents/workspace/InvoiceSystem3/WebContent/PdfTemp/BlankInvoice.pdf"); // input PDF
PdfStamper stamper = new PdfStamper(reader,
    new FileOutputStream("/Users/sluekeharris/Documents/Edited.pdf")); // output PDF
BaseFont bf = BaseFont.createFont(
    BaseFont.HELVETICA, BaseFont.CP1252, BaseFont.NOT_EMBEDDED); // set font

//loop on pages (1-based)
// for (int i=1; i<=reader.getNumberOfPages(); i++){

    // get object for writing over the existing content;
    // you can also use getUnderContent for writing in the bottom layer
    PdfContentByte over = stamper.getOverContent(1);

    // write username --> will be changed to organisation...
    over.beginText();
    over.setFontAndSize(bf, 20); // set font and size
    over.setTextMatrix(72, 720); // set x,y position (0,0 is at the bottom left)
    over.showText(user.getCompany()); // set text
    over.endText();

    //write username and address
    over.beginText();
    over.setFontAndSize(bf, 10); // set font and size
    over.setTextMatrix(460, 710); // set x,y position (0,0 is at the bottom left)
    over.showText(user.getUserName()); // set text
    over.setTextMatrix(460, 700);
    over.newLineShowText(user.getAddress());
    over.endText();

```

Figure 31

We can see that the `SetTextMargin()` contains 2 values that are the x and y co-ordinates on the PDF and a `ShowText()` which prints the String onto the document at the given location. To do this I had to use iText which is a paid subscription when using the service regularly. But for the propose of this project it did not cost anything.

Graphical User Interface Implementation

Below are some screenshots of what I regard the key pages in the application.



Welcome to the website!

If you are new to the site, check the Sign Up button to create an account 😊

Figure 32

Here is the first page the user is greeted with, it follows a similar style to Facebook's login page with the username and login in the top right hand side located in the menu bar. the user has two options in this. to either login with the username and password they provide or to create a new account by navigating to the signup page.

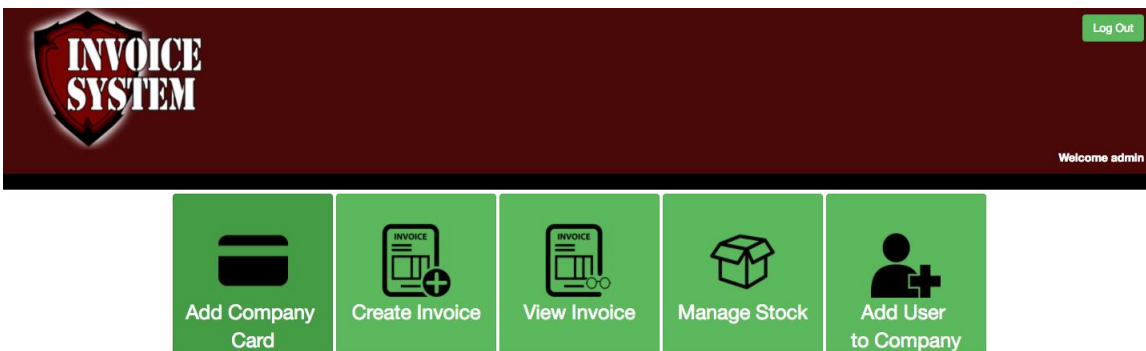


Figure 33

Above we can now see the main page a user will access when logging into their account. the user is greeted with a welcome message and then has five options they can pick from which allow to manage their account. Depending on the user's permissions they will be able to view and access different functions. From the screenshot above I made it easy to view and let the user assume what each function does just by looking at the button. there is also a logout feature which will end the user's session as a security measure.

Welcome admin

Subject Field:

Customer Details: **PO Number:**

Due Date:
(How Long In Days to pay the bill? E.G. 7 = 1 week)

Discount (if any):

Figure 34

The above page is one of the pages that takes time to fill out. I tried to shorten the amount of data the user has to enter by taking values from the user's account and calculating other data for the user as well. the fields that are marked with a "*" are mandatory fields that the user must enter some data in order to continue with the creation of the PDF invoice. once created the user can continue to the View Invoice section and click the button which will allow them to download the document.

Name	Category	Quantity	Cost	Date Made		
Velvet Shirt	Clothing	21	€ 23.0	2017-04-28	Update	Delete
Jeans	Clothing	34	€ 15.0	2017-04-28	Update	Delete
Chocolate Bar	Food	400	€ 2.0	2017-04-28	Update	Delete
iPad Mini	Electronics	12	€ 200.99	2017-04-28	Update	Delete
iPhone 7s	Electronics	20	€ 299.99	2017-04-29	Update	Delete
Crossbow	Adventure	12	€ 150.5	2017-05-02	Update	Delete
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		Add Stock	

Item	Quantity
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>

[Save Invoice](#)

Figure 35

The final page which I believe is important is the stock management section of the application which allows the user to create, read, update, delete stock from the user's company. if the user does not have the ability to update or delete stock the buttons will not appear for the user. also if the user does not have the ability to add stock the last line will not appear.

Testing

For this project prototype I'll be testing the software in two different ways. firstly, is security based. this is where I will try exploit my own site to find any vulnerabilities using the kali machine located on my computer. the second is a customer/consumer test. I will be testing the site on both experienced users of websites and inexperienced users. the second test will be assumed to not go as well as it could due to the minimalist design used in the program.

RESULTS

PENETRATION TEST RESULTS

For this test I have conducted multiple tests. For the first set of tests I have attempted to attack the website using KALI tools.

Two of the tools I'll be using are OWASP Zap which detects the most common problems found on websites and another tool called NIKTO which will also scan for vulnerabilities.

OWASP ZAP

On the right is a screenshot of the OWASP ZAP result of a hack against the webserver. Each of these vulnerabilities will be explained. It is better to run these tests and find out your weakness rather than not know them at all.

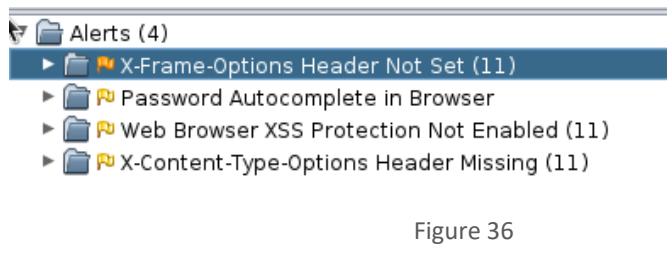


Figure 36

X-FRAME-OPTIONS

this means that the website is vulnerable to clickjacking by using iframes or similar techniques. this isn't a vulnerability to the system itself but more a vulnerability to the users. if a hacker was to use this iframe on their site, it could allow them to track the user's input and steal their information while they navigate the site. The way to block this is by not allowing other URL's accessing this site through an Iframe or similar.

PASSWORD AUTO COMPLETE

this is when the browser auto completes a password in the browser for the user. this is an issue as if another user was to login to the user's computer, they will have access to the users account. This is then a vulnerability set up by the user themselves rather than the system. We can set the forms and passwords to "AUTOCOMPLETE="off" "but when testing this against google chrome, you'll find that it doesn't actually work. as such the feature was added after testing but did not change the vulnerability.

WEB BROWSER XSS

this is probably one of the major vulnerabilities left out of the system. the website does not have user to user encryption set up at the time of the test. this means that when logging in and other functions are sent to the server. they are sent in unencrypted form leaving the possibility of attackers to intercept the package and possibly gain information about the account. This is to be implemented in a later version of the application and is the next step in the project security. as the

project is not publically available as of yet this is not as such a priority.

X-CONTENT-TYPE-OPTIONS HEADER MISSING

this is to do with some error handling issues with the site when passing multiple values through the servlet. the website started to return error 401, 403, 500 which return errors to the user and can lead to giving information about the SQL server to the user. Changes to the project have since been implemented following this test to prevent some of these errors from reaching the user.

One of the outcomes of this test shows us how the server can detect when a hack is in progress on the server and can

```
at java.sql.DriverManager.getConnection(DriverManager.java:671)
at backend.Backend.checkLogin(Backend.java:63)
Failed Login: 192.168.0.39
at controller.PresentationController.processLoginCheck(PresentationController.java:58)
at servlet.Servlet.loginCheck(Servlet.java:464)
at servlet.Servlet.doGet(Servlet.java:184)
at servlet.Servlet.doPost(Servlet.java:344)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:648)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:292)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)
at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:240)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:212)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:106)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:502)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:141)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:79)
at org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:616)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:88)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:522)
at org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java:1095)
at org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol.java:672)
at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1500)
at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.run(NioEndpoint.java:1456)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
at java.lang.Thread.run(Thread.java:745)
Backend - checkLogin - Exception3: User 'LoginUser' has exceeded the 'max_questions' resource (current value: 1200)
com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: User 'LoginUser' has exceeded the 'max_questions' resource (current value: 1200)
at sun.reflect.GeneratedConstructorAccessor60.newInstance(Unknown Source)
```

Figure 37

allow us to act appropriately in the given situation.

from the screenshot above, we can see that the server detected a failed login by the IP of 192.168.0.39. then when the user attempted to login over 1200 times as seen at the bottom, it blocked the user from trying to attempt again using another connection. This is most likely the reason of the X-Content-type-options header missing error we received during our tests.

NIKTO SCAN

```
- Nikto v2.1.6
-----
+ Target IP:          192.168.0.39
+ Target Hostname:    192.168.0.39
+ Target Port:        8080/JAXRSJsonCRUDEExample/
+ Start Time:         2017-05-03 19:35:57 (GMT-4)
-----
+ Server: Apache-Coyote/1.1
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS.
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS
+ OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients to save files on the web server.
+ OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server.
+ 7536 requests: 0 error(s) and 6 item(s) reported on remote host
+ End Time:          2017-05-03 19:36:12 (GMT-4) (15 seconds)
```

Figure 38

above is the results of an Nikto scan which is capable of finding other vulnerabilities as well as some found in the Zap.

Some of the similar ones found was no XSS, clickjacking, and X-Content-type-options header.

the other issue found was that the server accepted all GET, POST, PUT methods which may allow outside users to delete files and add files to the sever. Although this true in one sense as it is possible to send these requests, when the server receives these requests it ignores them due to no process present to deal with the request. so this is what we would call a false positive when using these testing tools.

CUSTOMER TEST RESULTS

For the customer tests let users run free amongst the website without any input from myself and these were the following results from the tests and tasks I had set out for the user to complete

Sarah Kadry Project supervisor - is very experienced using computers and has generally no issues with navigating user interfaces.

User 1 Stephen Harris- is a regular end level user. can navigate websites with ease and knows some general system shortcuts

User 2 Adam Butterly- is a college student studying computers, very experienced with navigating user interfaces

User 3 Ruth Doran - is a college student studying computers, very experienced with navigating user interfaces

User 4 Julie Harris- does not generally use computers so the user accessed the site through an iPad for these tests which upped the times significantly. User does not use the internet for other reasons other than social media on her mobile device or tablet.

Task: Create an account on the website

USER	Time Taken	Needed Assistance	Comments
Sara Kadry	1:31	none	Unclear what section was wrong when filling out form
USER 1	2:45	none	Password gave user issues
USER 2	1:47	none	No Issues to report
USER 3	2:01	none	Password gave issues, annoying the way user needed to re-type all fields
USER 4	5:23	Help Understanding tax number and what the section even meant	Was unsuccessful at navigating to sign up and needed extra assistance when signing up to the website on most of the sections after multiple explanations

From these results the next logical stage is to implement a proper warning to the user on which section that they are incorrectly entering and then following this sending the values back to the user that were entered correctly back to the page so these do not have to be re-entered again.

The next task was to create an Invoice. this was a trick task as such as the user will have to first figure out that they need to add stock to their account to make an invoice.

USER	Time Taken	Needed Assistance	Comments
Sara Kadry	2:49	No	The site gave back sufficient notice about what to do when first setting up an account
USER 1	3:34	No	"that's grand"
USER 2	3:36	No	n/a
USER 3	2:13	No	n/a
USER 4	n/a	Yes	User did not understand what button lead the user to add stock. and then needed extra assistance to create another invoice

From this we can see that there are some improvements to the message that we can send to the user. for example, the user was informed to add stock, but not where this is located. Following this vie added that to add stock the user should click the “Manage Stock” button.

Finally, the last challenge set to the users was to set up another user under the company account with restrictions of their choosing

USER	Time Taken	Needed Assistance	Comments
Sara Kadry	2:34	no	Password Gave issues like in the sign up section, more info about password rules needed
USER 1	3:12	no	Password gave issues
USER 2	1:52	no	Completed with ease
USER 3	3:52	no	n/a
USER 4	5:56	yes	had issues understanding what the purpose of this action was and did not want other users logging in under her password. Had to explain that this was a new password for a pretend employee to log into.

Although the last user struggled with most of the tasks, we can actually learn the most from this user. yet again some of the sections can be easily misinterpreted when not explained sufficiently at each section. Changes across the entire website to make it more user friendly were implemented in order to make these core processes less confusing to these users.

The user for this test that I am most interested in was USER 2 as he is the sort of user I’ll be targeting my application at. This user does run a business and manages stock intake and output. Some comments that I expected from the user such as adding more than six items to the invoices were expressed as well as an ability for a customer to pay the user through the website and or for him to pay suppliers of companies using the website. These are all future prospects of the project and will be implemented at a later stage.

Evaluation

After receiving the results from both the penetration test and the reviews from the Front end testers, I have come to separate conclusions on both Security and end user experience.

From the results of the Penetration Test we can conclude that the web application is secure against the most common forms of attacks that most websites are not protected against today. Obviously the results are clearly not perfect. Some improvements with SSL and preventing Clickjacking can further protect users from their information being leaked to an external source, but for the purpose of this project, the project is highly secured.

I've given an overall security vulnerability score as "low" as a result from the tests. No System is perfectly secure but when a penetration test cannot find an immediate vulnerability it can be assumed that the system is more than adequate. The main features I would like to implement is SSL and fix the remaining vulnerabilities.

The End-User testing returned some useful information as well, we can clearly see that there are some areas for improvement. Some big areas like an overhaul on design and some smaller areas just redirecting the user to the correct area to perform a task. Some area which I would have liked to improve on is when the user enters incorrect information, the user should not have to re-enter information on the entire page.

Conclusions

In its current state, the project does not offer as much functionality as some bigger rivals that are currently on the market. In most cases I've researched the web applications allow users to create invoices and accept payment from users who can log on and pay via card. This is obviously a major draw back to my application as this sort of functionality could be key to gain any market value.

What these other big companies do not do is manage stock intake and output. I have set the ground work in this application to be set up as a major stock and invoice manager. Upon future work we can have more sophisticated systems around the front end users so that it will be appealing to users to join my site over others like adding stock to the system by entering the amount of stock taken in and having it automatically added to the system.

On the security side of the project I believe I have done a very good job at securing the users information. There is obviously some areas which need improving like the SSH encryption to protect the users data traveling across the network but for the most part I have secured the users information.

The project itself has the opportunity to become a fully working and professional looking invoice system that is competitive against today's existing companies. With new ways to hack computers arising every day, new security measures need to be implemented. This means that the project itself has a forever raising ceiling about the security limitations of the project. There are currently hardware limitations set to the project as well as the entire project needs to be able to run off my computer for the purpose of this project. But if we can design the project to make money we can then set up the project to go live which will then be able to make revenue to pay for any extra costs that may occur when expanding the hardware processing power or memory size.

Future Prospects

The Project itself has some major improvements and updates that will improve the overall performance and security of the project. The Goal of this project is to create a watertight project with no data leakage. without the use of SSL, or some other form of certificate, this is impossible, if a user uses this service on an insecure network they may be victim an account breach affecting their account confidentiality.

Some other features that didn't make the final cut of this prototype include is password recovery, two step verification or multi factor verification. the original idea was to include a mobile app that would unlock your account via a fingerprint scan. this became to become to time consuming taking over two weeks to get any sort of an app that would send a request to unlock any given account. even still this data was sent insecurely. for these reasons the idea is not implemented.

Using a real credit card verification system proved to be costly, and when the credit card does not perform any major functions I settled with creating my own credit card verification system that simply returns a true or false string depending on whether or not the credit card number is the correct length. In the future I plan to expand this functionality by allowing users to make transactions between accounts and allow payments to their own banks by offering the user to enter their bank details. A solid foundation has been set for this to become a reality.

The end-user functionality is not the worst but not the best. The name of the site itself is less than imaginative and the logo looks cheap. one of the pages that time was spent on is the Main page after logging into the site. my goal would be to make the remainder of the site just as easily accessible and visual as this page.

Furthermore, although we have already a lot of server side validation, some client side validation wouldn't go amiss either. this wouldn't as so much improve the security of the entire system but just assist the users when entering any data to the site. instead of having to almost assume that their password is valid, I could force a regular user to enter a correct password length or correct quantity value before pressing submit on the page. this would save a lot of grief for the end user as if some information is missing on some of the pages, the page requires you to fill out all of the information again in its current state.

Appendix

References & Other Materials Used

Multinational Companies with leaked Data - <https://haveibeenpwned.com/PwnedWebsites>

iText - <http://developers.itextpdf.com/> : Used to add text to pdf document

OWASP - <https://www.owasp.org/>

Kali Penetration test Machine - <https://www.kali.org/downloads/>

Stack Overflow - <https://stackoverflow.com/> this was used through out the entire project to help with some trivial Java problems to even some more advanced ones.

AES Encryption Help - <http://aesencryption.net/> gave source code for the AES which I re-proposed to fit my project

W3Schools - <https://www.w3schools.com/> helped with some of the more trivial HTML and SQL issues in the project.

Project Proposal

Objectives

The Company I plan to do my 4th year project on is called Thriftify. At the moment Thriftily has an online books section where charities can go online and sell books easily through amazon. The website currently has no security features to stop anyone from hacking the website as it is only a start-up company.

For this project I will be securing the sensitive information on the website which is currently located at "<http://www.zonesoftware.ie/thriftify/index.jsp>". I will be building an Invoice system that will allow companies who use our software to be able to view an invoice of how much money is owed to them from Thriftify. Currently the Charity we are working with is called the NCBI (National Council for the Blind Ireland), but this is soon planned to expand further to St. Vincent De Paul.

This Invoice system is important because it allows Thriftify and Charity companies to view how much we owe them. The information would be calculated from the existing database and be available to view on the website in a new section. The sensitive data of the invoice will include the bank details of where we are sending the money and how much we are sending. This data should not be publically available to anyone who is not authorised to see it.

The main objective of this project is to secure the data surrounding the invoice system. The security features like Multiple Level authentication system, database encryption and Website Encryption will be implemented into the security system.

I intend on using AES (Advanced Encryption Standard) to secure the data on the website as it is seen to protect against all forms of attack aside from brute force. It can use all encryption keys in the 128, 192 or 256-bit cypher. The 192 and 256 being used for heavy duty encryption purposes which I intend on doing.

For the Multiple level authentication, I plan on using a text message and/or fingerprint scanner to allow users to access the invoice, while also needing their password to log into their account.

Background

Thriftify is a company with a charitable status. The overall objective of Thriftify is to bring charity shops to the digital age. Most of the products in a charity shop are selling for a fraction of the price that they could be sold online for. Thriftify books plan is to sell books that are donated to charity shops through amazon for a short term goal. In the long term Thriftify plan on moving onto other goods like clothing and electronics. But due to books having a unique identifier for every book, we are starting with books.

The software works by a user who is employed by the charity. The user scans a book into our system using a barcode scanner and after an exchange of information with amazon's web service, the book is then either declared eligible to be uploaded or not based on profits made. If the book is not profitable it is then discarded to a recycling company who offer €60 per tonne or the book is sold on the shop floor for €1-€2 on average. If the book is saleable it is then uploaded to amazon at a fair, competitive, price. The mark-up on books sold online is huge at over 200% compared to books sold in the shop after commission and other costs, and over 1400% if the book is sent to recycling. When the book is sold online,

the shop is notified through our software and the book gets sent from the shop.

Money made from the book is then sent to Thriftify who take 30% commission after calculating costs of sending the book. The remainder of the money is sent back to the charity companies on an annual basis.

The software currently has different level users ranging from CEO's to Basic users. CEO's have access to view all shops under their company. This means they can change permissions that each user has, view analytics on each and all shops under their company name, create new users and shops. This is very much a Role based access control system. The basic idea of the system is that if you have the permission to view a shop, you can view analytics on the shop. The higher up you are the more shops you can see. Another function is also being able to give other users permission to view shops that they cannot, provided that you have access to their shops, essentially upgrading their accounts. Basic accounts only have the ability to scan books through our system and get info on where to send them.

My project idea is to build an invoice system that is accessible by the top CEO accounts provided they do not want to share the privilege with other accounts. This singular functionality will be surrounded by multiple security techniques to protect the information efficiently. This Project is being used in a real world scenario so by the end of the project I plan to have it running live on the web link provided above.

SQL injection is always going to be a problem. Recently the Multinational company Sony was hacked using SQL injection compromising 1,000,000 accounts. In March 2008, 134 million credit cards were hacked. 2010 millions of personal emails were hacked through Epsilon. The list of hacks goes on. My project plan is to stop this from happening and is a main reason I picked this project.

Technical Approach

Reading through articles online it is evident that SQL injections and other forms of malicious attackers are very common on the internet as apparent from above. Although not published the infamous website "ClickandGo.ie" website was recently a victim of ransom ware. All of their files were encrypted to which Click & Go had to pay out a large lump sum to get the decryption key. These are the sort of attack I plan on securing against. And is the source of my inspiration for this project although it is a different company. Data Confidentiality is a major problem in today's computing as evident from the

The project will be running locally on my laptop using Apache Tomcat server until the project build is finished. After finishing I will send the code off to the software developer in Thriftify who will then integrate the code into the live version of the website.

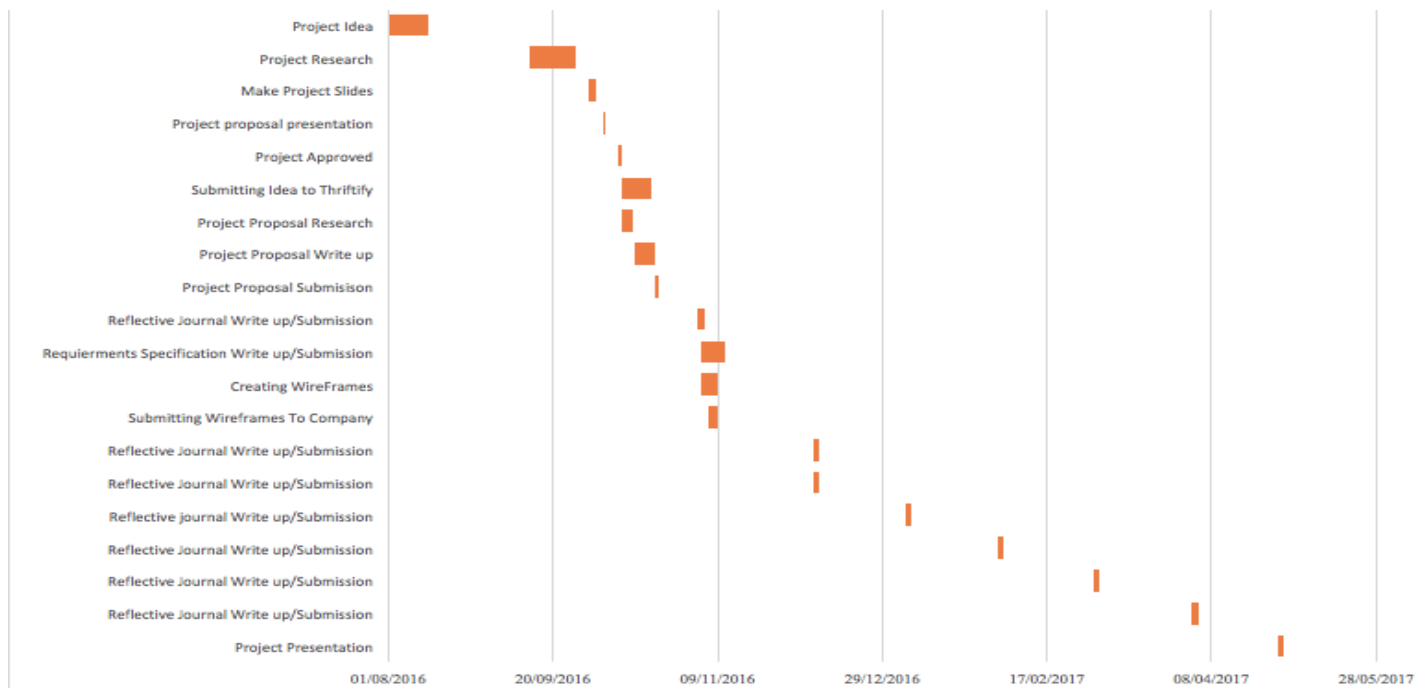
There are many different solutions to encrypting your database online including TDE, HSM etc. over the coming weeks I will be researching the best method of encrypting the Thriftify Database in order to deter attackers from our website.

I intend to display my invoices using Bootstrap with a separate downloadable .pdf file for printing physical copies. It will contain an Address of Thriftify Offices, Contact number, Billing address, number of sales, vat. etc. and other headings you would typically see on an Invoice.

Special resources required

The system will be running live on a server paid by for the company. Ill be using a phone with a finger print scanner to identify any users when doing the Multi-Factor identification. No other special recourses required.

Project Plan



Task	Start-Date	Duration	End-Date
Project Presentation	28/04/2017	2	20/04/2017
Reflective Journal Write up/Submission	02/04/2017	2	04/04/2017
Reflective Journal Write up/Submission	03/03/2017	2	05/03/2017
Reflective Journal Write up/Submission	02/02/2017	2	04/02/2017
Reflective journal Write up/Submission	05/01/2017	2	06/01/2017
Reflective Journal Write up/Submission	08/12/2016	2	09/12/2016
Reflective Journal Write up/Submission	08/12/2016	2	09/12/2016
Submitting Wireframes To Company	06/11/2016	3	09/11/2016
Creating WireFrames	04/11/2016	5	09/11/2016
Requirments Specification Write up/Sub	04/11/2016	7	11/11/2016
Reflective Journal Write up/Submission	03/11/2016	2	04/11/2016
Project Proposal Submission	21/10/2016	1	21/10/2016
Project Proposal Write up	15/10/2016	6	21/10/2016
Project Proposal Research	11/10/2016	3	14/10/2016
Submitting Idea to Thriftify	11/10/2016	9	20/10/2016
Project Approved	10/10/2016	1	10/10/2016
Project proposal presentation	05/10/2016	1	05/10/2016
Make Project Slides	01/10/2016	2	03/10/2016
Project Research	13/09/2016	14	27/08/2016
Project Idea	01/08/2016	12	12/08/2016
Building Project		On going	
Project Analsys and Design		On going	
Project Build		On going	
Supervisor Meeting/s		4	
Mid Point Presentation		2	
Project Submission		1	
Project Showcase		1	

Technical Details

The Software is built on Eclipse in a dynamic web project and android studio. The languages for creating the software include Java SE 8 on an Apache Tomcat server v8. We are using Html 5 and Boot Camp for our base design with some JavaScript to display the information to the user in clever ways.

The Libraries we are using include:

Amazonservices.jar, activation.jar, commons.jar, jacksons.jar, Mysql-connecor.jar, servlet-api.jar, sqljdbc.jar, jaxb.jar and jdom.jar.

The company has its own library on the live system called Thriftify.jar as well

Evaluation

I will be evaluating the system live on customers who use the system on a daily basis in their work to ensure the system is on par with there needs. I will then follow up with the feedback given and adjust to their needs and wants accordingly while keeping the systems security a priority.

The feedback on the new update will be taken through phone. After using the system for a set period, the phone numbers of each shop in Ireland is called and feedback from each one, if possible, is taken. This will be done towards the end of the project when the system is live with a few minor adjustments added or taken away before the final upload if needed. This allows a direct line of communication with the customers to gain some real feedback on the application.

I will also get some feedback through Thriftify CEO and software developers throughout the project and on the final product.

I will also try to ethically hack the website using a number of techniques the most prominent being SQL injection to test my security for weak points with the objective being that I fail to gain any information at all.

I believe using all of these techniques will give me a valid evaluation of my work and give feedback on where I should improve weather it be design work and ease of use or security. Reflective Journal's

Reflective Journals

Reflective Journal for September

My Achievements

This month included researching the few ideas I had for the final year project, I contacted my Security lecturer (Sara Kari) for advise on what would be a better idea for my final year project.

One idea seems very bulky and involves creating a website that integrates fraud detection on credit/debit cards to minimise false purchases. The algorithm would detect after a few checks if the card is outside a standard deviation and mark the card as fraudulent. After three attempts the card would be banned from being used again on the website. I am waiting on feedback to see if this revolves around my security class enough and is a sufficient idea.

The next idea seems a little bit underwhelming, the idea was to secure the website I was working on during my internship. The internship was in a start up company and has not got a lot of security. So I thought it might be a good idea to use that opportunity as a project as I know it revolves around my course.

Both ideas have been marginally researched. Waiting for some feedback on both to see which is a better option or if I should do more research into another idea.

Update:

I have spoken with Sara again after having my idea rejected and we have cleared up what ill be looking to do with my project. The plan is now to go to Thriftify with my project ideas and hope they will accept them.

My Reflection

I feel I should have researched a bit more in preparation for the project. I still feel blind going into this project. After discussing with Sara I believe the better option is to work with the second idea of securing the website and keep scaling the idea as I work. I'm starting to learn some theory on security while in college and already thinking of a few ideas on how to create work on my project.

Getting some advice and pointed in the right direction is a huge help. Security is a new area for me so a little nudge in the right direction is very helpful.

Intended Changes

Next month I intend to do some extensive research on my project to meet requirements that need to be reached for the final year project. I feel confident I can create enough work.

I plan to be more organised with my work now that I have a solid starting point in the assumption my idea for the company is accepted.

Supervisor Meetings

Date of Meeting: Mon 10th Oct

Items discussed: Project Idea/Plan

Action Items: Contacted Thriftify about project ideas. Am confident that they will accept my idea.

Reflective Journal for October

My Achievements

This month I have refined my original idea. Although the plan was to work with Thriftify, after many emails sent and not many received I have planned to go off on a different route and find open source code online where I can implement my original plan on or to develop a small sized dynamic web project like before with Thriftify. using the latter method would stop any confusion between what my work is and what work was already done.

My Reflection

This Month I continued to do more research on my idea. I have plans on using AES encryption on my database and have refined my idea even further. But I still feel like I should be setting aside more time to work on the project compared to other work I'm doing for college. Need to set out a weekly schedule to properly organise myself.

Other than the work related issues I feel I am making some progress in relation to my project. I am more confident after my supervisor meetings about my project. Before when I was emailing Thriftify I was getting stressed as they were very uncooperative, where in the beginning they seemed very happy to allow me to work on the website.

Intended Changes

For this month I intend on creating a weekly schedule to organise myself properly and store it on my phone. This way I can set aside time to work on my project. I also plan to secure exactly what project I'll be working on with in the next few days. I will then re-edit my project proposal and finish off my Requirements Spec. this way I will then have an end goal to work towards

Supervisor Meetings

Date of Meeting: Mon 27^h Oct

Items discussed: Requirements Spec

Action Items: Research open source ideas for my project.

Reflective Journal for November

My Achievements

This Week was slow due to the amount of projects needed to be done. but I have made progress on refining my idea further. I can now start working on a prototype to show something for the presentation on Dec 14th. I got my schedule under control like I suggested last month and am working in time to work on projects and research ideas during my job working hours to take the load of work off during the week. Next semester I plan on having a better time schedule to get massive amounts of work done to finish my project with time left over.

My Reflection

Stress of other projects is overwhelming so as soon as they are done ill be able to focus properly on the main project here. other than that I have done some reports and researched code to use with my project.

Intended Changes

I plan to work later nights the next 2 weeks to spare some time for project work to make sure I am not falling behind.

Supervisor Meetings

Date of Meeting: 3rd Nov 2016

Items discussed: technical report

Action Items: Start Putting together my technical report.

Date of Meeting: 10th Nov 2016

Items discussed: technical report revision

Action Items: looked at what else is needed in the tech report

Date of Meeting: 17th Nov 2016

Items discussed: technical report / prototype

Action Items: start development of the prototype and continue with the tech report

Date of Meeting: 24th Nov 2016

Items discussed: technical report / prototype

Action Items: finalise the technical report and have prototype presentable for the mid point presentation.

Reflective Journal for December

My Achievements

This month i have greatly improved my workload in relation to my project. I have created my API in which my mobile app will connect to, i have the beginnings of a encryption class created, and my database design is al but finished bar a few tweaks that will need to be done along the process of creating my application. For the next week i plan to finish off my encryption class so it is working as intended and create a API key so it is secure. This will hopefully not be a huge task to complete as i already have an idea on how to solve this issue.

My Reflection

The first couple of weeks was slow in relation to the project update but i am now in full swing doing substantial amounts of work on the project staying back late and working a lot faster.

Intended Changes

I intend on keeping my current work rate up and not slow down.

Supervisor Meetings

Date of Meeting: 10th of Fed, 3rd Feb

Items discussed: Project and Documents

Action Items: Logic behind API keys and Encryption Keys and other development needs of the project.

Reflective Journal for January

My Achievements

This month was the month of the mid-point presentation. For this i needed a prototype. So i began work and got together a working login with session management. I also did a small bit of Web design for the prototype so it could be seen the general colour scheme that i was going for. Unfortunately i was having major issues connecting to the database so it was not possible to show off that. I also got a working Database design for the project which gave me a huge insight from the mid-point on how i can improve the project. Ideas such as approvers would allow another layer of security.

My Reflection

I believe my work was substantial this month and i still haven't really stopped working with all the exams currently on. I will resume work on the project after the exam and get back up to speed on the project shortly after the exams.

Intended Changes

I plan to make the necessary changes to the database to add security to the users who use the system. This will overall improve my project for such a small change. Maybe set limits for each user on how much they can order? So you can only assign someone else to order the same or less than you if you have permission.

Supervisor Meetings

Date of Meeting: Due to exams, there has been few

Items discussed: Prototype

Action Items: create a prototype and have presentation slides ready

Reflective Journal for February

My Achievements

For this month, i worked mainly on functionality. this involved getting all the pieces working together. the encryption utility now works with login passwords, phone numbers and an array of other methods. Also began work on the credit card validation which takes in a credit card number and checks if it is valid. This is an example of how it'll be done through a third party software like BOI or Visa. for this project tho we will just check if the card is 16 digits long after sending it to a API i have made and returning a true or false value. i want to further secure this by adding encryption to the request.

Update:

After speaking with sara we are looking to leave aside some of the more complex features and start working on the core security features of the project which will add more value to the final project.

My Reflection

This Project has become more beefed out than expected. a few features i originally intended to add at the start of the project might be added towards the end depending on what time i have left for the project

Intended Changes

Add some more security features that will improve the core functionality of the application.

Supervisor Meetings

Date of Meeting: Mon 10th Feb

Items discussed: Project Idea/Plan

Action Items: The general improvement of core security features for my application is the general topic. So working on encrypting the sensitive data using high level encryption.

Reflective Journal for March

My Achievements

For the final month before the final upload i worked on solidifying the core Security functionality. the application can now send encrypted value to each other. you can now create an account, add a credit card and check if it is a valid number. i now want to add the functionality of uploading a PDF document though a default layout. it will add graphics. i have prevented SQL injection with logging on by comparing a return value with a

My Reflection

Need to keep up to date with project paper work, secondly i want to add the functionality of adding a company under a new user's name and allowing them to create users with less privileges. then add the stock amounts and allowing to add/subtract stock from a menu.

Intended Changes

More core security and functionality.

Supervisor Meetings

Date of Meeting: Tue 2nd 9th 16th 23th

Items discussed: Project development

Action Items: working on bringing the whole project together, steps on creating the project functionality are becoming painfully repetitive on the system but are necessary for completing the project.

Declaration Cover Sheet for Project Submission

SECTION 1 Student to complete

Name: Stephen Harris
Student ID:X13507947
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 (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: Stephen. H. Date: Sun 7 May 2017

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Complete the sections above and attach it to the front of one of the copies of your assignment,