



Secure Software Development

Year 2 (2020/21), Semester 3

SCHOOL OF INFOCOMM TECHNOLOGY

Diploma in Cybersecurity & Digital Forensics

Diploma in Information Technology

Assignment

Secure Web Application

Duration: Weeks 8 - 16
Weightage: 50%
Individual/Team/Both: Both
Deadline: **9 Aug 2020, 11:59pm (Sunday)**

Group Name	Team02 SSD: Sale of Toys
Student IDs	S10194982A, S10196470C, S10196678G, S10197294
Student Names	Ezra Ho Jincheng, Gladys Chua, Lim Shu Zheng, Xie Zhuo Han
Module Group	SSD

Contents

Project Overview	6
Project Vision	6
Project Description	6
Project Features	6
Shared Components of the features	8
I. Secure Software Design: Threat Modelling	8
Phase 1 – 3-Tier Application Architecture.....	8
○ Diagram Topology and Design	8
Feature 1: CRUD Database and Authorization	9
1.1 Secure Software Requirements.....	9
1.1.1 Use Case and Misuse Case Modeling.....	9
1.1.2 Data Labeling	10
1.1.3 Data Classification	10
1.1.4 Subject / Object Matrix.....	15
1.1.5 Surveys (Questionnaires & Interviews).....	15
1.1.7 Relevant Security Concepts and Management	19
1.2 Secure Software Design	22
1.2.1 Security Design Implemented.....	22
1.2.2 Attack Surface Evaluation	25
1.2 Threat Modeling.....	26
1.3.2 Data Access Control Matrix.....	27
○ Technologies used to build the application.....	27
○ External dependencies	27
1.3 Identifying, Prioritizing and Implementing Controls.....	31
1.5 Documentation and Validation	32
1.6 Secure Software Processes.....	34
1.7 Secure Software Coding.....	35
1.7.1. Defensive Coding Implementations	35
1.8 Secure Software Testing	38
1.8.1 Test Strategy	38
1.8.2 Testing Plan.....	40
1.9 Security Conclusion of the Feature.....	45

2.1.1 Use Case and Misuse Case Modeling (done)	46
2.1.2 Data Classifications (done)	46
2.1.3 Data Labelling (done)	47
2.1.4 Subject/Object Matrix.....	49
2.1.5 Surveys (need 1 more question then finish alrd)	49
2.1.6 Relevant Security Concepts.....	51
2.2 Secure Software Design	53
2.2.1 Feature related security design.....	53
2.2.2 Attack Surface Evaluation	54
2.3 Secure Software Coding.....	57
2.3.1 Feature related Defensive Coding Practices	57
2.4 Secure Software Testing	59
2.4.1 Test Strategy	59
2.4.3 Test Cases	61
2.5 Security Conclusion of the feature	62
Feature 3: Authentication	63
3.1 Secure Software Requirement	63
3.1.1 Use case and Misuse Case Modeling.....	63
Feature 4: Audit	77
4.1 Secure Software Requirements	77
4.1.1 Use Case and Misuse Case Modeling.....	77
4.1.2.1 Data Labelling.....	77
4.1.2.2 Data Classification	78
4.1.3 Subject/Object Matrix.....	79
4.1.4 Surveys (Questionnaires and Interviews).....	79
4.1.4.2 Who should be allowed to view the log?	79
4.1.4.3 What details should the audit log show?	80
4.1.5 Related Security Requirements	81
4.1.5.1 CIA.....	81
4.1.5.2 Session Management.....	82
4.1.5.3 Error Management.....	82
4.2 Secure Software Design.....	83
4.2.1 Related Security Design Principles	83

4.2.2 Attack Surface Evaluation	83
4.2.3 Threat Modeling.....	85
4.3 Secure Software Coding.....	87
4.4 Secure Software Testing	88
4.5 Security Conclusion of the Feature.....	90
Feature 5: Miscellaneous Functions (Search, Shopping Cart, Payment, Error Page).....	91
5.1 Secure Software Requirements.....	91
5.1.1 Use Case and Misuse Case Modeling (done)	91
5.1.2 Data Classification & Labelling (done)	92
5.1.4 Data Ownership (done).....	93
5.1.5 Surveys (Questionnaires and Interviews)	93
5.1.6 Relevant Security Concepts and Management	95
5.2 Secure Software Design	98
5.2.1 Secure Software Design Principles.....	98
5.2.2 Attack Surface Evaluation	99
5.3 Threat Modeling	101
5.3.1 Subjects & Objects of the system	101
5.3.2 Data Access Control Matrix.....	101
○ Technologies used to build the application.....	101
○ External dependencies	102
5.3.3 Trust Boundaries.....	102
5.3.4 Entry Points	102
5.3.5 Exit Points	103
5.3.6 General data flow	103
5.3.7 Privileged functionality.....	104
5.3.8 Mis-Actors	104
5.3.9 Potential and applicable threats through STRIDE.....	105
5.4 Identifying, Prioritizing and Implementing Controls	106
5.4.1 Delphi Ranking	106
5.4.2 Probability x Impact (P x I) Ranking.....	106
5.5 Documentation and Validation	107
5.6 Secure Software Processes	109
5.6.1 Versioning	109

5.6.2 Code Analysis	109
5.6.3 Code & Peer Review	110
5.7 Secure Software Coding (editing)	110
5.7.1 Defensive Coding Practices	110
5.8 Secure Software Testing	113
5.8.1 Test Strategy	113
5.8.2 Test Plan	114
5.8.3 Test Cases	117
5.9 Security Conclusion of the Feature	118
6.1 Secure Software Requirements	119
6.1.1 Use Case and Misuse Case Modeling	119
6.1.2.1 Data Labelling	119
6.1.2.2 Data Classification	120
6.1.3 Subject/Object Matrix	120
6.1.4 Surveys (Questionnaires and Interviews)	120
6.1.4.1 Who are allowed to see the review page?	121
6.1.4.2 Who can create reviews?	121
6.1.4.3 Are they allowed to edit and delete their own reviews?	122
6.1.5 Related Security Requirements	123
6.1.5.1 CIA	123
6.1.5.2 Session Management	124
6.1.5.3 Error Management	124
6.2 Secure Software Design	125
6.2.1 Related Security Design Principles	125
6.2.2 Attack Surface Evaluation	126
6.2.3 Threat Modeling	127
6.3 Secure Software Coding	130
6.4 Secure Software Testing	131
6.5 Security Conclusion of the Feature	133

Project Overview

Project Vision

Our product vision is to facilitate online sale of toys through an online website in a secure manner.

Project Description

Our service is to simply prevent attackers from gaining access into the backend of the website, which could crumble the entire security of the website.

Project Features

1. Registration & Login

This is to uniquely identify each user upon a successful registration and login. The Registration function allows for the use to register an account with the application, while the Login function is used to authenticate registered accounts to grant them greater access and control within the application.

2. CRUD Databases

This is created to Create, Read, Update, and Delete products in the database. This only allows authorized users (such as the Owner) to modify the necessary product information within the database.

3. Role-Based Authorization

Role-Based Authentication is the approach to restricting system access to authorized users only. It also includes permissions to modify and operate on objects such as Create, Read, Update or Delete as these are the defined responsibilities and authority of an authorized user.

4. Authentication

This is the process of identifying the users to be who they say they are. This can come in many forms such as 2-Factor Authentication, Short Message Service (SMS) or Email verification. Some also have a Single Sign On (SSO) service on the browser. All of these are deployed such that the applications can properly verify the person trying to gain access into the account and application is the same person that has registered, and not anyone else.

5. Audit Log

This records the happenings of the application in the system. This would include the timestamp of the accessed object, what was modified, who modified it. It would enable Owner to detect unauthorized changes to the application. One of two scenarios would be an unauthorized user making an unauthorized change, or an authorized user making an unauthorized change. Both scenarios are integrity violations.

Work Distribution

1. CRUD Databases, Search Function, Customer Account Page, 2-Factor Authentication, External Login (Google, Facebook, Microsoft), Email Confirmation, Report Feature 1, Report Feature 5– Ezra Ho
2. Authentication, Role-Based Authorization, Shopping Cart, Payment Page, Report Feature 3– Zhuo Han (please check the amount of code, I did not push much but I did the code in large chunks and helped my teammates with multiple coding problems)
3. Audit, Review Section, Website Design, Report Feature 4, Report Feature 6 – Gladys
4. Registration and Login, Cross site scripting (XSS), Report Feature 2 – Shu Zheng

Shared Components of the features

I. Secure Software Design: Threat Modelling

Security Objectives:

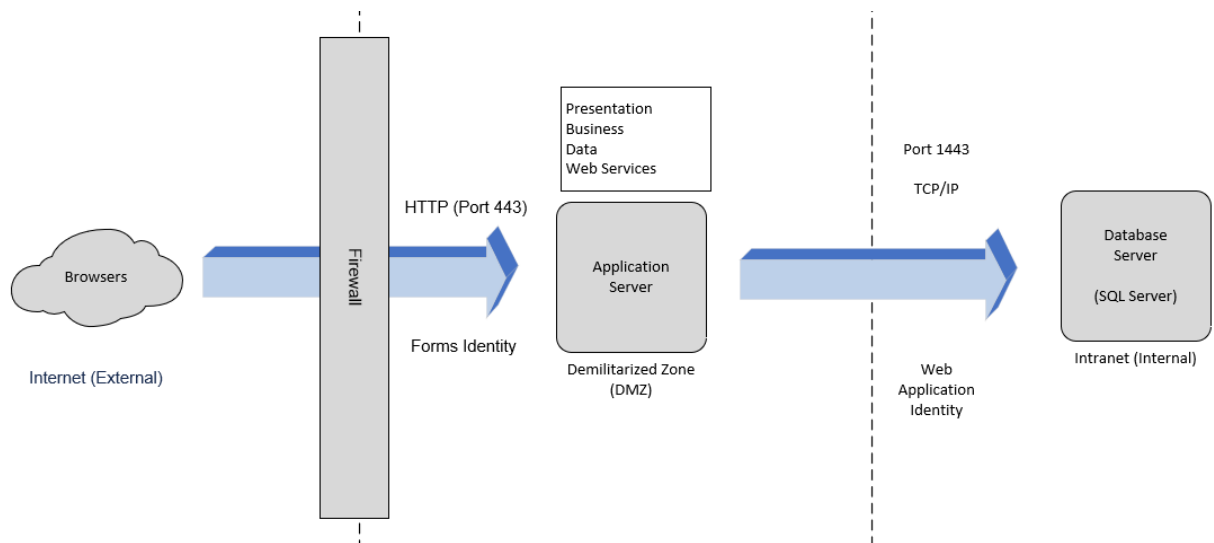
Objective 1: To provide a secure service to customers.

Objective 2: To provide an uninterrupted and seamless service to customers.

Objective 3: To provide a wide range of toys for children of all ages.

Phase 1 – 3-Tier Application Architecture

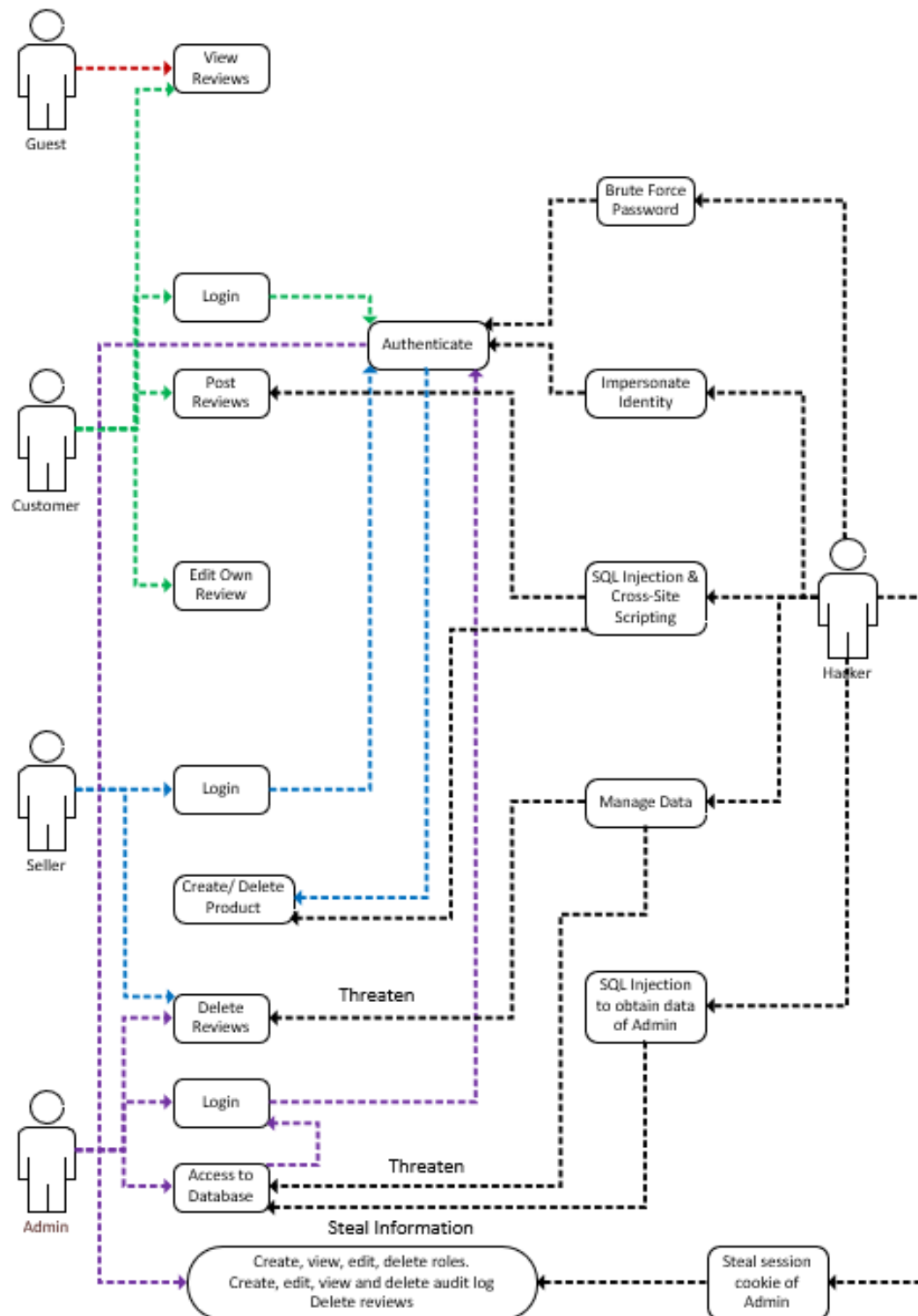
- Diagram Topology and Design



Feature 1: CRUD Database and Authorization

1.1 Secure Software Requirements

1.1.1 Use Case and Misuse Case Modeling



1.1.2 Data Labeling

The product page, which consists of product listings, is viewable by the public. This allows users to view the products and reviews to decide whether to make a purchase.

Data	Impact when loss of		
Toys Table	Confidentiality	Integrity	Availability
ID	Confidential	High	Highly Essential
Toy_Name	Public	Medium	Moderately Essential
Age	Public	Low	Non-Essential
Price	Public	High	Highly Essential
Category	Public	Low	Moderately Essential
ToyImage	Public	Low	Moderately Essential

1.1.3 Data Classification

- Structured Database

All the data which are required to be present in the database are stored within the database in a structured manner. By storing the data in a database, this allows the data to be more easily find by the owner.

```

1 CREATE TABLE [dbo].[Toy] (
2     [ID] INT IDENTITY (1, 1) NOT NULL,
3     [Category] NVARCHAR (60) NOT NULL,
4     [Age] NVARCHAR (60) NOT NULL,
5     [Price] DECIMAL (18, 2) NOT NULL,
6     [Toy_Name] NVARCHAR (60) NOT NULL,
7     [ToyImage] NVARCHAR (MAX) NULL,
8     CONSTRAINT [PK_Toy] PRIMARY KEY CLUSTERED ([ID] ASC)
9 );

```

Toy (ID, Category, Age, Price, Toy_Name, ToyImage)

```
1 CREATE TABLE [dbo].[Customers] (  
2     [CustomerID] INT          IDENTITY (1, 1) NOT NULL,  
3     [Username]   NVARCHAR (MAX) NULL,  
4     [Name]       NVARCHAR (MAX) NULL,  
5     [Email]      NVARCHAR (MAX) NULL,  
6     [Address]    NVARCHAR (MAX) NULL,  
7     [Age]        INT          NOT NULL,  
8     [PhoneNum]   NVARCHAR (MAX) NULL,  
9     [Comment]    NVARCHAR (MAX) NULL,  
10    CONSTRAINT [PK_Customers] PRIMARY KEY CLUSTERED ([CustomerID] ASC)  
11 );
```

Customer (CustomerID, Username, Name, Email, Address, Age, PhoneNum, Comment)

```
1 CREATE TABLE [dbo].[AuditRecords] (  
2     [Audit_ID]      INT          IDENTITY (1, 1) NOT NULL,  
3     [AuditActionType] NVARCHAR (MAX) NULL,  
4     [Username]      NVARCHAR (MAX) NULL,  
5     [DateTimeStamp] DATETIME2 (7) NOT NULL,  
6     [KeyListingFieldID] INT      NULL,  
7     [RoleName]      NVARCHAR (20) NULL,  
8     CONSTRAINT [PK_AuditRecords] PRIMARY KEY CLUSTERED ([Audit_ID] ASC)  
9 );
```

AuditRecords (AuditID, AuditActionType, Username, DateTimeStamp, KeyListingFieldID)

```
1 CREATE TABLE [dbo].[AspNetUserTokens] (  
2     [UserId]      NVARCHAR (450) NOT NULL,  
3     [LoginProvider] NVARCHAR (450) NOT NULL,  
4     [Name]        NVARCHAR (450) NOT NULL,  
5     [Value]       NVARCHAR (MAX) NULL,  
6     CONSTRAINT [PK_AspNetUserTokens] PRIMARY KEY CLUSTERED ([UserId] ASC, [LoginProvider] ASC, [Name] ASC),  
7     CONSTRAINT [FK_AspNetUserTokens_AspNetUsers_UserId] FOREIGN KEY ([UserId]) REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE  
8 );
```

UserTokens (UserId, LoginProvider, Name, Value)

```

1 CREATE TABLE [dbo].[AspNetUserRoles] (
2     [UserId] NVARCHAR (450) NOT NULL,
3     [RoleId] NVARCHAR (450) NOT NULL,
4     CONSTRAINT [PK_AspNetUserRoles] PRIMARY KEY CLUSTERED ([UserId] ASC, [RoleId] ASC),
5     CONSTRAINT [FK_AspNetUserRoles_AspNetRoles_RoleId] FOREIGN KEY ([RoleId]) REFERENCES [dbo].[AspNetRoles] ([Id]) ON DELETE CASCADE,
6     CONSTRAINT [FK_AspNetUserRoles_AspNetUsers_UserId] FOREIGN KEY ([UserId]) REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE
7 );
8
9
10 GO
11 CREATE NONCLUSTERED INDEX [IX_AspNetUserRoles_RoleId]
12 ON [dbo].[AspNetUserRoles]([RoleId] ASC);

```

UserRoles (<UserId>,<RoleId>)

```

1 CREATE TABLE [dbo].[AspNetUsers] (
2     [Id] NVARCHAR (450) NOT NULL,
3     [UserName] NVARCHAR (256) NULL,
4     [NormalizedUserName] NVARCHAR (256) NULL,
5     [Email] NVARCHAR (256) NULL,
6     [NormalizedEmail] NVARCHAR (256) NULL,
7     [EmailConfirmed] BIT NOT NULL,
8     [PasswordHash] NVARCHAR (MAX) NULL,
9     [SecurityStamp] NVARCHAR (MAX) NULL,
10    [ConcurrencyStamp] NVARCHAR (MAX) NULL,
11    [PhoneNumber] NVARCHAR (MAX) NULL,
12    [PhoneNumberConfirmed] BIT NOT NULL,
13    [TwoFactorEnabled] BIT NOT NULL,
14    [LockoutEnd] DATETIMEOFFSET (7) NULL,
15    [LockoutEnabled] BIT NOT NULL,
16    [AccessFailedCount] INT NOT NULL,
17    [FullName] NVARCHAR (MAX) NULL,
18    [BirthDate] DATETIME2 (7) NOT NULL,
19    [Age] INT NOT NULL,
20    CONSTRAINT [PK_AspNetUsers] PRIMARY KEY CLUSTERED ([Id] ASC)
21 );
22
23
24 GO
25 CREATE NONCLUSTERED INDEX [EmailIndex]
26 ON [dbo].[AspNetUsers]([NormalizedEmail] ASC);
27
28
29 GO
30 CREATE UNIQUE NONCLUSTERED INDEX [UserNameIndex]
31 ON [dbo].[AspNetUsers]([NormalizedUserName] ASC) WHERE ([NormalizedUserName] IS NOT NULL);

```

Users (Id, UserName, NormalizedUserName, Email, NormalizedEmail, EmailConfirmed, PasswordHash, SecurityStamp, ConcurrencyStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnabled, LockoutEnd, LockoutEnabled, AccessFailedCount, FullName, BirthDate, Age)

```
1 CREATE TABLE [dbo].[AspNetUserLogins] (  
2     [LoginProvider] NVARCHAR (450) NOT NULL,  
3     [ProviderKey] NVARCHAR (450) NOT NULL,  
4     [ProviderDisplayName] NVARCHAR (MAX) NULL,  
5     [UserId] NVARCHAR (450) NOT NULL,  
6     CONSTRAINT [PK_AspNetUserLogins] PRIMARY KEY CLUSTERED ([LoginProvider] ASC, [ProviderKey] ASC),  
7     CONSTRAINT [FK_AspNetUserLogins_AspNetUsers_UserId] FOREIGN KEY ([UserId]) REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE  
8 );  
9  
10  
11 GO  
12 CREATE NONCLUSTERED INDEX [IX_AspNetUserLogins_UserId]  
13 ON [dbo].[AspNetUserLogins]([UserId] ASC);
```

UserLogins (LoginProvider, ProviderKey, ProviderDisplayName, <UserId>)

```
1 CREATE TABLE [dbo].[AspNetUserClaims] (  
2     [Id] INT IDENTITY (1, 1) NOT NULL,  
3     [UserId] NVARCHAR (450) NOT NULL,  
4     [ClaimType] NVARCHAR (MAX) NULL,  
5     [ClaimValue] NVARCHAR (MAX) NULL,  
6     CONSTRAINT [PK_AspNetUserClaims] PRIMARY KEY CLUSTERED ([Id] ASC),  
7     CONSTRAINT [FK_AspNetUserClaims_AspNetUsers_UserId] FOREIGN KEY ([UserId]) REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE  
8 );  
9  
10  
11 GO  
12 CREATE NONCLUSTERED INDEX [IX_AspNetUserClaims_UserId]  
13 ON [dbo].[AspNetUserClaims]([UserId] ASC);
```

UserClaims (Id, <UserId>, ClaimType, ClaimValue)

```
1 CREATE TABLE [dbo].[AspNetRoles] (  
2     [Id] NVARCHAR (450) NOT NULL,  
3     [Name] NVARCHAR (256) NULL,  
4     [NormalizedName] NVARCHAR (256) NULL,  
5     [ConcurrencyStamp] NVARCHAR (MAX) NULL,  
6     [CreatedDate] DATETIME2 (7) DEFAULT ('0001-01-01T00:00:00.0000000') NOT NULL,  
7     [Description] NVARCHAR (MAX) NULL,  
8     [IPAddress] NVARCHAR (MAX) NULL,  
9     CONSTRAINT [PK_AspNetRoles] PRIMARY KEY CLUSTERED ([Id] ASC)  
10 );  
11  
12  
13 GO  
14 CREATE UNIQUE NONCLUSTERED INDEX [RoleNameIndex]  
15 ON [dbo].[AspNetRoles]([NormalizedName] ASC) WHERE ([NormalizedName] IS NOT NULL);
```

Roles (Id, Name, NormalizedName, ConcurrencyStamp, CreatedDate, Description, IPAddress)

```
1 CREATE TABLE [dbo].[AspNetRoleClaims] (
2     [Id] INT IDENTITY (1, 1) NOT NULL,
3     [RoleId] NVARCHAR (450) NOT NULL,
4     [ClaimType] NVARCHAR (MAX) NULL,
5     [ClaimValue] NVARCHAR (MAX) NULL,
6     CONSTRAINT [PK_AspNetRoleClaims] PRIMARY KEY CLUSTERED ([Id] ASC),
7     CONSTRAINT [FK_AspNetRoleClaims_AspNetRoles_RoleId] FOREIGN KEY ([RoleId]) REFERENCES [dbo].[AspNetRoles] ([Id]) ON DELETE CASCADE
8 );
9
10
11 GO
12 CREATE NONCLUSTERED INDEX [IX_AspNetRoleClaims_RoleId]
13 ON [dbo].[AspNetRoleClaims] ([RoleId] ASC);
```

RolesClaims (Id, RoleId, ClaimType, ClaimValue)

```
1 CREATE TABLE [dbo].[RatingNReviews] (
2     [Rating_ID] INT IDENTITY (1, 1) NOT NULL,
3     [Review] NVARCHAR (MAX) NULL,
4     [Usernam] NVARCHAR (MAX) NULL,
5     [RatingStar] INT NULL,
6     [DateReview] DATETIME2 (7) NOT NULL,
7     [ListID] INT NOT NULL,
8     [ImageToy] NVARCHAR (MAX) NULL,
9     CONSTRAINT [PK_RatingNReviews] PRIMARY KEY CLUSTERED ([Rating_ID] ASC)
10 );
```

RatingNReview (Rating_ID, Review, Usernam, RatingStar, DateReview, ListID, ImageToy)

```
1 CREATE TABLE [dbo].[PaymentDetails] (
2     [cusid] INT IDENTITY (1, 1) NOT NULL,
3     [ShippingAddress] NVARCHAR (MAX) NOT NULL,
4     [PostalCode] NVARCHAR (MAX) NOT NULL,
5     [Token] NVARCHAR (MAX) NULL,
6     [Total] FLOAT (53) NOT NULL,
7     [Email] NVARCHAR (MAX) NULL,
8     CONSTRAINT [PK_PaymentDetails] PRIMARY KEY CLUSTERED ([cusid] ASC)
9 );
```

PaymentDetails (cusid, ShippingAddress, PostalCode, Token, Total, Email)

1.1.4 Subject / Object Matrix

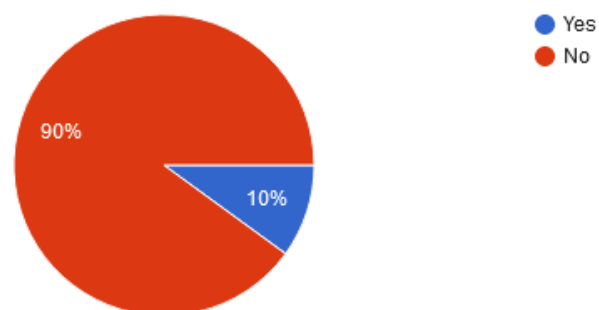
Due to the role-based authorization assigned to each user, this allocates each user to have respective responsibilities regarding the product page. The guest and customer have similar authorization levels accessing the database when it comes to the creation, deletion, modification, and readability of the products.

Data	Objects			
Toys Table	Guest	Customer	Seller	Owner
ID	-	-	RD	RD
Toy_Name	R	R	CRUD	RD
Age	R	R	CRUD	RD
Price	R	R	CRUD	RD
Category	R	R	CRUD	RD
ToyImage	R	R	CRUD	RD

1.1.5 Surveys (Questionnaires & Interviews)

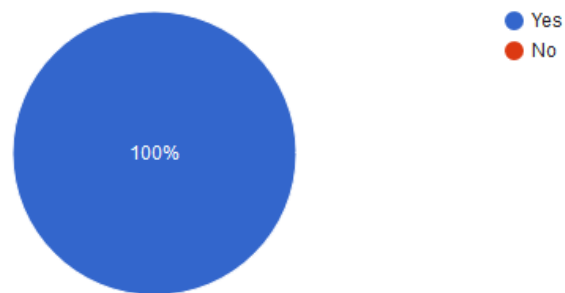
- Should Guests (unregistered user) be allowed to add a product to shopping cart?

10 responses



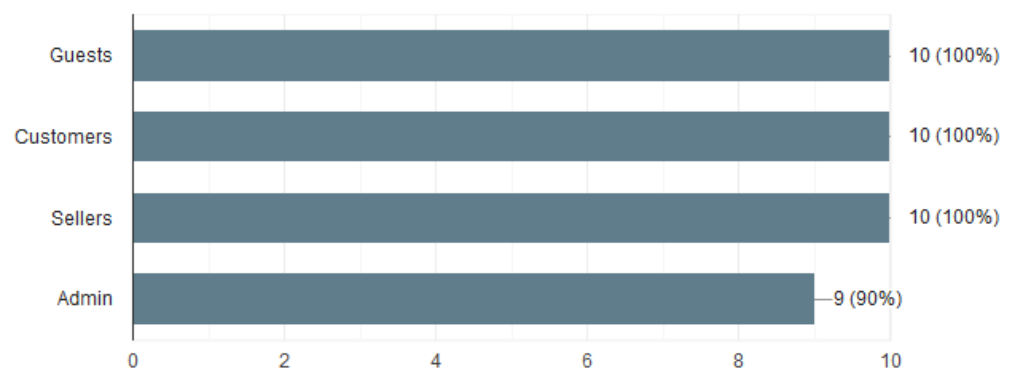
- Should Customers (registered user) be allowed to add a product to shopping cart?

10 responses



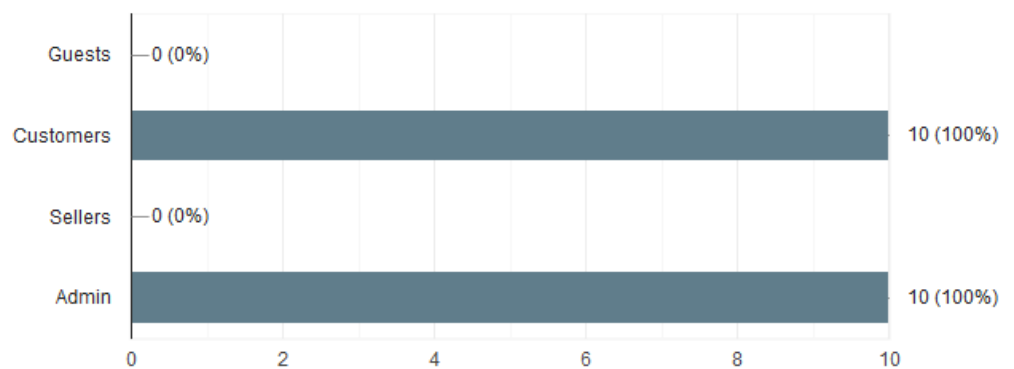
- Which roles should have access to the reviews of a product?

10 responses



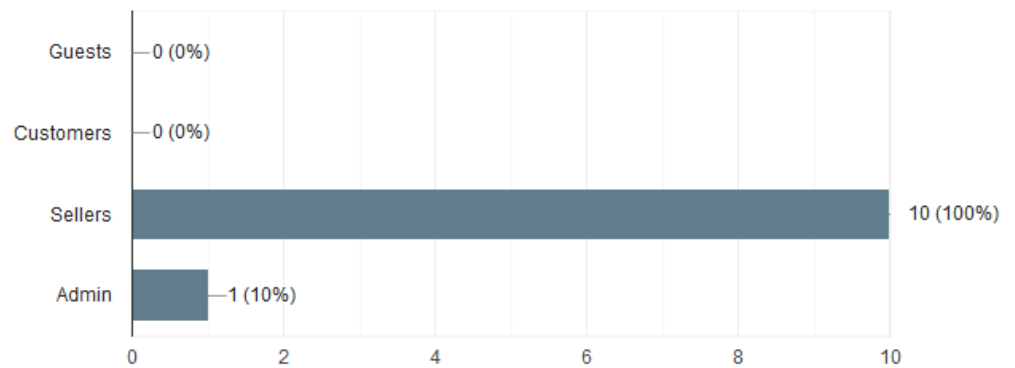
- Which role can remove the review of a product?

10 responses



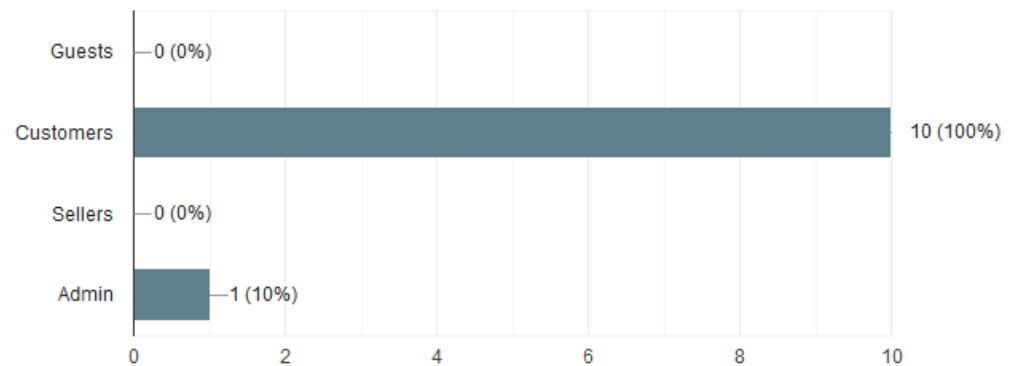
- Which role can edit the price of the product?

10 responses



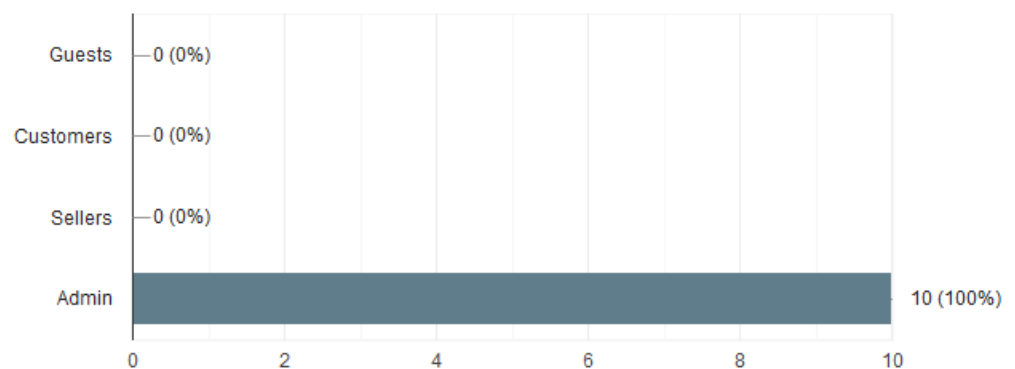
- Which role is able to modify the review of a product?

10 responses



- Which role is able to access the database of the website application?

10 responses



a. What can the software do with the data?

- i. The software can process, transmit and store the data that is entered as input on the website.

b. What types of data can the database store?

- i. The types of information are roles of users, reviews, customer information, product information, payment information, and audit log.

c. What type of information is obtained and stored in the database?

Information stored in the database are:

ID	UserName	NormalisedUserName
Email	NormalizedEmail	EmailConfirmed
PasswordHash	SecurityStamp	ConcurrencyStamp
PhoneNumber	PhoneNumberConfirmed	TwoFactorAuthentication
LockoutEnd	LockoutEnabled	AccessFailedCount
FullName	BirthDate	Age

d. What rights does each of the roles have?

- i. The guests are only able to view the products. It is not able to add product into cart without being logged in.
- ii. The registered and logged in customers can view products, create reviews of purchased products, delete reviews of previously purchased products, add products into shopping cart, add credit card information into the application, review "My Accounts" Page and make necessary changes.
- iii. The sellers can view products, add products into their own store, edit the price of their products, delete their products from their store.
- iv. The owner can view products, review purchased products, delete products that are deemed offensive, delete reviews if deemed offensive, assign user roles and access database.

e. How confidential should the database be to the public users?

The database of the website application should remain highly confidential and strictly accessible to the owner. This database access allows the owner to assign roles, remove users from the website application and other administrative roles.

1.1.7 Relevant Security Concepts and Management

- **C.I.A.**

- **Confidentiality**

- Information inside the database should always remain confidential to all parties except the owner. This is to maintain confidentiality of information as only the owner has the authority to view and manage the database.
 - The reviews of the products page are not confidential since it must be viewed by other roles (Guests, Customer, Sellers, and Owner).

- **Integrity**

- Integrity of input is critically important in the review. This is because if it happens to be modified without consent or authorization, the people who sees the review might decide to change their decision based on the review. If this happens, the business of the store will eventually decrease as lesser people will purchase the product and hence the store will decrease in sales due to the negative reviews seen on the product page review section. Hence, the integrity of reviews is critically important and should be held with high confidentiality.

- **Availability**
 - The database is always available to the owner to have full control to create, delete and modify the data within the database.
 - Sellers are strictly not allowed to view database. Credit card information and payment related processes are handled by owner.
 - Customers can modify their credentials by entering their personal details, this will update their personal information which is reflected on the database. Customers are unable to view or modify personal information of other users.
 - Guests can only view the products and their reviews within the database, it is unable to modify information within the database as it is not a registered user.
- **A.A.A.**
 - **Authentication**
 - When seeing reviews, the roles do not need to sign in to view it as they are unable to edit the review unless they are the user who created that review.
 - Only a customer, which is a registered guest, can create and edit a review of a product which was once purchased by the user.
 - Seller and Owner can view the product review. Only the owner can delete the review if deemed offensive. Seller can strictly only view the review.

- **Authorization**

- All roles are given access to reading the review since the confidentiality level is low. But, the authorization of creating, deleting, reading, and updating reviews are only granted to roles which are given by the owner.
- Database authorization is only provided to the owner. The owner can enter the database to modify the database. No other roles have such authority. If other roles have such authority, the database has a chance of being ruined and unstructured as there are many “admins”, causing data collision since two “admins” might attempt to change the same data at the same time. Another security issue is that the “admins” may leak out private information when they have access to the database. This can cause huge damage for the company in terms of reputation and reliability, and also the damage done to users who have their data stored on the database as hackers can easily obtain information now that their credentials are leaked in public.
- Customer does not have authority to make any changes. Customers have the authority to create, modify reviews. Sellers does not have the ability to create, modify and delete reviews. The owner has the authority to delete reviews if deemed offensive. The owner is able to create or modify reviews as it is allowed to purchase the products.
- Hence, it is especially critical to carefully authorize people into roles as a small mistake of accidentally assigning a user as an “admin” can cause huge damage.

- **Session Management**

- Customers are required to authenticate themselves by registering and logging in before they can create or edit a review. This is to ensure that the person who is attempting to modify have the right authority to make such a change. Otherwise, if roles are not authenticated then “sellers” are able to modify users’ product reviews and customers are able to delete sellers’ products because of no distinct roles.

- **Error & Exception Management**

- Access is denied to customers who attempt to edit or delete a product.
- Access is denied to sellers attempting to edit a review.
- When an error occurs, the page should not list the details of the type of error to prevent hackers from taking advantage of this error, causing damage to the site using SQL Injection or Cross-Site Scripting. Instead, an error page should be displayed such that unauthorized roles are not able to view the information which are not viewable to them.

1.2 Secure Software Design

1.2.1 Security Design Implemented

- **Least Privileges**

- Different roles are given different levels of privileges, which directly corresponds to their level of authorization.
- Guests are given the least level of privilege, as it is only authorized to view the products and reviews.
- Next in line are the customers, which have more privileges but is still quite basic as they can only create, edit, delete their own reviews, and read reviews of others. They can also add and remove items into shopping cart and make payment for items.

- Sellers have some privileges which can be considered on par with customers. They can create, edit, delete their own product listing. They are permitted to view reviews, but not permitted to delete reviews that were written by customers.
 - The owner has the highest authority as it can access all the database; create, modify, delete, and read the database as well as manage roles. It can also add, delete item from shopping cart and make payment. Thus, shows that it has the highest privileges while guests have the least privileges.
- Fail Safe
 - An access denied page will be shown to all roles on the actions that they are not allowed to perform due to authorization privileges given by the owner.
- Economy of Mechanism
 - The pages are all built in a simple manner that allows users to view them in a comfortable manner. A design that is simpler design will mean that security implementation will be easier and checking through each function would be much more thorough. This will reduce the overall attack surface for the attacker, giving them a harder time to break the security of the website. All the user inputs are all placed in one database, this is to increase the security as only the security of only one database needs to be enhanced.

- Psychological Acceptability
 - The security mechanisms available on the website application are all there because they are a necessity to make the site secure from attacks. There is no added hassle in the steps at all.
 - Sellers can easily create, delete, modify their products if they please if they have registered and authenticated themselves. Owner must require authentication before accessing the website application and the database.
 - All of these are not to increase the difficulty for the users to enter the website, but these are all added as a security feature for a safer and more pleasant experience for all the users who access this website.
- Leveraging Existing Components
 - All the features used on this website are all tested for compatibility. Some of the codes are used from Microsoft and others are from the internet. The project is definitive proof that the code functions as works as intended to, such as the regular expressions for inputs and review functions. All of these ensure that the attack surface does not expand, and the website will remain secure.

1.2.2 Attack Surface Evaluation

All surfaces are included here because all their inputs will eventually be processed into the database. Hence, all attack surface evaluation is shown here first.

Surface	Is it exploitable by the hacker?	Mitigation to an attack by hacker.
Search Function	Yes. The input boxes allow hackers to input malicious code into the function.	Input validation and restriction on certain characters in the input boxes, prevents SQL Injection and Cross-site Scripting attacks.
Payment Page	Yes. The input boxes for payment allows hackers to input malicious code into the function.	Input validation and restriction on certain characters in the input field prevents SQL Injection and Cross-site scripting attacks.
Shopping Cart Page	No. This page only allows users to view the item and does not have any input fields.	-
My Account Page	Yes. This page allows some input by the user such as email address.	Input validation and restrictions have been set to prevent certain characters in the input field to prevent SQL Injection and Cross-site scripting attack.
OTP Page (not yet)	Yes. The OTP textbox allows input of malicious code.	Have strong input validation and validate all user inputs and filter for characters and SQL content
Product (Toy) Page	Yes. The creation of products allows for hackers to input malicious codes.	Input validation and restriction on certain characters in the input field prevents SQL Injection and Cross-site scripting.

Registration Page	Yes. The registration page has input fields which allow hackers to input malicious codes.	Input validation and restriction on certain characters and symbols in the input field prevents SQL Injection and Cross-site scripting.
Login Page	Yes. The login page has input fields which allows hackers to input malicious codes.	Implementing input validation, restricting certain characters and symbols as well as having input sanitization on the input field prevents SQL Injection and Cross-site scripting.
Error Page	No. There are no input fields in the error page for hackers to input malicious codes.	-

1.2 Threat Modeling

1.3.1 Subjects & Objects of the system

- Guests can strictly only view products and its description.
- Customers can view products, create, and edit review, modify their personal information on their "My Accounts" page, add item(s) to their shopping cart as customers have been given minimum amounts of authorization.
- Seller can strictly only create, delete, modify products and have no access to database as authorized by owner.
- Owner can manage products, view, or delete feedback, strictly view audit logs as authorized.

1.3.2 Data Access Control Matrix

	Guest	Customer	Seller	Owner
Customer Data	-	C R U D	-	R
Product Data	R	R	C R U D	R D
Cart Data	-	C R U D	-	C R U D
Payment Data	-	C R	-	C R

- **Technologies used to build the application**

- Visual Studio
- SQL Server
- ASP.NET, Razor Pages

- **External dependencies**

- None

1.3.3 Trust Boundaries

- It exists between external (Internet) and Demilitarized Zone (DMZ) Network.
- It exists between Demilitarized Zone (DMZ) Network and Internal (Intranet) Zones.

1.3.4 Entry Points

Entry Point	Description of Entry Point	Accessibility limits
Port 443	Users can connect to the online	Guests, Customers,

	platform via port 443 (HTTPS)	Sellers, Owner
--	-------------------------------	----------------

1.3.5 Exit Points

Exit Point	Description of Exit Points	Role Accessibility
HTTPS Port 443	Users connect to the platform via port 443 (HTTPS)	Guests, Customers, Sellers, Owner
Index and Details Page (Welcome Page, Toys Page, Role Page, Audit Log)	These pages display their own information to the viewers, respectively. This allows clear separation of intention for each page.	All pages - Guests, Customers, Sellers, Owner Role Page, Audit Log - Only Owner
Search Results Page	This page returns the results of the keywords searched by the user to filter the toys.	Guests, Customers, Sellers, Owner

1.3.6 General data flow

- Guests
 - Guest browses the product page
 - Guest views details of an item
- Customers
 - An existing customer logs in □ Customer views products
 - Customer logs in □ Customer adds an
 - Customer logs in à Views details of an item
- Seller
 - Seller registers for an account □ Owner assigns seller to a seller authorization
 - Seller logs in □ Creates an item
 - Seller logs in □ Updates an item
 - Seller logs in □ Deletes an item
 - Seller logs in □ Views details of an item
 - Seller logs in □ View review an item on Review Page
- Owner
 - Owner logs in □ Owner view audit logs
 - Owner logs in □ Owner goes to Manage Roles Page

1.3.7 Privileged functionality

- The codes that allow the increase in privileges or execution of operations are identified.
- Guests have functions such as product management (read) are identified.
- Customers has product management (read), account management (create, update, delete) and feedback management (create, update, delete) functions identified.
- The seller functions consist of product management (read, create, edit, delete) are identified.

- The owner has functions which consists of product management (read), account management (create, update and delete), feedback management (create, update, delete) as well as audit management (read) are identified.

1.3.8 Mis-Actors

- Internal and external threats and identified and introduced.
 - Examples of Human Mis-actors: External hacker, Hacktivist group, Rouge administrator
 - Examples of Non-human Mis-actors: Internal running process making unauthorized changes, malware.

1.3.9 Potential and applicable threats through STRIDE

STRIDE	Security Property	Identified threat
Spoofing	Authentication	<ul style="list-style-type: none"> • Brute Force Authentication • Cookie Replay • Session Hijacking • CSRF
Tampering	Integrity	<ul style="list-style-type: none"> • Cross-Site Scripting (XSS) • SQL Injection
Repudiation	Non-repudiation	<ul style="list-style-type: none"> • Audit Log Manipulation • Insecure Backup
Information Disclosure	Confidentiality	<ul style="list-style-type: none"> • Eavesdropping Verbose Exception • Output Caching • Path Traversal

Denial of Service	Availability	<ul style="list-style-type: none"> Website Defacement
Elevation of Privilege	Authorization	<ul style="list-style-type: none"> Logic Flaw

1.3 Identifying, Prioritizing and Implementing Controls

1.4.1 Delphi Ranking

Threat	D	R	E	A	DI	Total	Result
SQL Injection	3	3	2	3	2	13	High
Cross Site Scripting	3	3	3	3	3	15	High
Cookie Replay	3	2	2	2	2	11	Medium
Session Hijacking	2	2	2	1	3	10	Medium
Cross Site Request Forgery	3	1	1	1	1	7	Low
Verbose Exception	2	1	2	3	1	9	Medium
Brute Force	2	1	1	3	2	9	Medium
Eavesdropping	2	1	1	2	2	8	Medium
Insecure Backup	2	1	1	2	2	8	Medium
Audit Log Deletion	1	1	1	1	3	7	Low
Output Caching	3	3	2	3	3	14	High
Website Defacement	3	2	1	3	3	12	High
Logic Flaws	3	1	1	2	1	8	Medium

1.4.2 Probability x Impact (P x I) Ranking

Threat	Probability of Occurrence (P)			Business Impact		P	I	Risk
	R	E	DI	D	A	(R + E + DI)	(D + A)	
SQL Injection	3	2	2	3	3	7	6	42
Cross-Site Scripting	3	3	3	3	3	9	6	54
Cookie Replay	2	2	2	3	2	6	5	30
Session Hijacking	2	2	3	2	1	7	3	21
CSRF	1	1	1	3	1	3	4	12
Verbose Exception	1	2	1	2	3	4	5	20
Brute Force	1	1	2	2	3	4	5	20
Eavesdropping	1	2	2	2	2	5	4	20
Insecure Backup	1	1	2	1	2	4	3	12
Audit Log Deletion	1	1	3	1	1	5	2	10
Output Caching	3	2	3	3	3	8	6	48
Website Defacement	2	1	3	3	3	6	6	36
Logic Flaws	1	1	1	3	2	3	5	15

1.5 Documentation and Validation

After documentation of the lists of threats and controls and the residual risks involved, our group conclusively validated that the platform threat model would have the following key characteristics:

- The 3-tier application architecture model is accurate and is up to date.
- That threats are identified when crossing each trust boundary and each data element.
- Ensure that the design built is secure and is able to withstand attacks.

1.5.1 Threat Mitigation

This table shows the PI Threat level of each vulnerability and some ways to mitigate the attacks.

Threat - (P x I Rank)	Steps to mitigate threat
SQL Injection - (42)	<ul style="list-style-type: none">• Input validation.• Parameterized Queries.• Stored Procedures.• Escaping.
Cross Site Scripting - (54)	<ul style="list-style-type: none">• Escaping.• Input validation.• Sanitization.
Cookie Replay - (30)	<ul style="list-style-type: none">• Cookie-less authentication.• Encrypt cookies to avoid tampering.• One-time token.
Session Hijacking - (21)	<ul style="list-style-type: none">• Generate and use random session identifiers.• Abandons session after use.• Log off automatically after leaving the site.

	<ul style="list-style-type: none"> • Using “httpOnly” in the code to disable JavaScript code from reading the session identifier. • Setting the secure flag.
CSRF - (12)	<ul style="list-style-type: none"> • Use unique session tokens. • Use referrer origin checks. • Complete mediation. • Implement anti-CSRF token. • Use SameSite flag in cookies.
Verbose Exception - (20)	<ul style="list-style-type: none"> • Use non-verbose error message. • Fail secure.
Brute Force - (20)	<ul style="list-style-type: none"> • Don't allow weak passwords using regular expression. • Balance psychological acceptability with strong passwords.
Eavesdropping - (20)	<ul style="list-style-type: none"> • Enable data encryption. • Sniffers detection. • Disallow rogue systems.
Insecure Backup - (12)	<ul style="list-style-type: none"> • Enable data encryption. • Using SSL (Secure Socket Layer) for transport or IPsec (Internet Protocol Security) for network in-transit protection.
Audit Log Deletion - (10)	<ul style="list-style-type: none"> • Don't allow other users direct access to the database
Output Caching - (48)	<ul style="list-style-type: none"> • Don't cache credentials • Implement complete mediation
Website Defacement - (36)	<ul style="list-style-type: none"> • Load balancing and DR

	<ul style="list-style-type: none"> • Disallow URL redirection
Logic Flaws - (15)	<ul style="list-style-type: none"> • Design Reviews

- Each threat has been explicitly considered and controls for mitigation, acceptance, or avoidance have been identified and mapped to the threats they address.
- The residual risk of that threat is determined and formally accepted by the business owner, if the decision to accept risk is made.
- It is also important to revalidate the threat model, should the scope and attributes of the application change.

1.6 Secure Software Processes

1.6.1 Versioning

- This ensure that all members on the team is working on the most up-to-date version of the code, granting rollback if necessary. This has a secondary function, which is the audit log, displaying the changes in code and the users who pushed the changes. We coded on our individual devices and used Bitbucket as a platform to compile and update our codes for others.

1.6.2 Static Code Analysis

- This is the process of inspecting the code in Visual Studio 2019 to figure out which codes may have the potential to cause errors. When codes are committed and pushed, the codes that are pulled by the teammates might cause errors due to incompatibility issues. The errors are indicated by a red underline in Visual Studio.

1.6.3 Dynamic Code Analysis

- Dynamic inspecting involves inspecting the code during execution to figure out if there are any errors present. One example of an error is “build error”, where why the project can compile, but has issues when it tries to run. This is critical to know the errors that are still present in the project, ensuring that it can run as expected us users. The reason for these errors is perhaps certain parameters within the database not defined.

1.6.4 Code & Peer Review

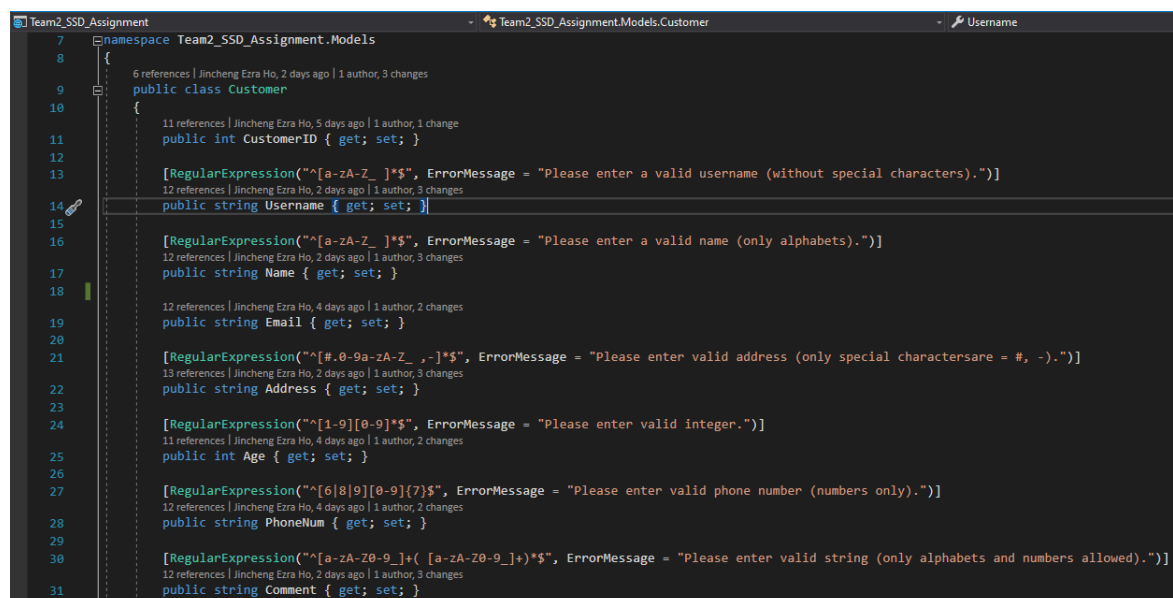
- Code is tested and reviewed by other members in the team to highlight any mistakes and to correct it before proceeding further. These are not to point out mistakes but learning points to improve the application when such a mistake occurs. Ultimately, it lowers the vulnerability and errors caused by the application when it is finished.

1.7 Secure Software Coding

1.7.1. Defensive Coding Implementations

- Input Validation**

Across the entire website application, all the possible attack points, especially where users can input texts, are guarded by regular expressions as a form of input validation. This is to prevent any malicious attacks to the website such as SQL Injection or Cross-Site Scripting. The input validation prevents all special characters (e.g. #, \$, %, ^, &, *, (,), !, @, _, -, +, =, /, \, :, ;, ", ', <, >, `, ~, , , .). It only allows integers from 0 to 9, upper-case and lower-case alphabets and spaces. These ensure that all the inputs the user enters are verified and authenticated. Different inputs require different conditions and cases, but across the entire website except for the email address, special characters are not allowed. Email address boxes will only allow special characters such as @, _ (underscore) and . (fullstop).



```

7 namespace Team2_SSD_Assignment.Models
8 {
9     6 references | Jincheng Ezra Ho, 2 days ago | 1 author, 3 changes
10     public class Customer
11     {
12         11 references | Jincheng Ezra Ho, 5 days ago | 1 author, 1 change
13         public int CustomerID { get; set; }
14
15         [RegularExpression("[a-zA-Z_]*$", ErrorMessage = "Please enter a valid username (without special characters).")]
16         public string Username { get; set; }
17
18         [RegularExpression("[a-zA-Z_]*$", ErrorMessage = "Please enter a valid name (only alphabets).")]
19         public string Name { get; set; }
20
21         12 references | Jincheng Ezra Ho, 2 days ago | 1 author, 3 changes
22         public string Email { get; set; }
23
24         12 references | Jincheng Ezra Ho, 4 days ago | 1 author, 2 changes
25         [RegularExpression("[#0-9a-zA-Z_,-]*$", ErrorMessage = "Please enter valid address (only special characters are #, -).")]
26         public string Address { get; set; }
27
28         [RegularExpression("[1-9][0-9]*$", ErrorMessage = "Please enter valid integer.")]
29         public int Age { get; set; }
30
31         11 references | Jincheng Ezra Ho, 4 days ago | 1 author, 2 changes
32         [RegularExpression("[689][0-9]{7}$", ErrorMessage = "Please enter valid phone number (numbers only).")]
33         public string PhoneNum { get; set; }
34
35         12 references | Jincheng Ezra Ho, 2 days ago | 1 author, 3 changes
36         [RegularExpression("[a-zA-Z0-9_]+([a-zA-Z0-9_]+)*$", ErrorMessage = "Please enter valid string (only alphabets and numbers allowed).")]
37         public string Comment { get; set; }
38     }
39 }
  
```

Create

Customer

Username

Terius2 #

Please enter a valid username (without special characters).

Name

Ezra %

Please enter a valid name (only alphabets).

Email

ezra1@gmail.com

Address

Jurong *

Please enter valid address (only special characters are = #, -).

Age

18 ah

Please enter a valid number.

PhoneNum

*

Please enter valid phone number (numbers only).

Comment

This ()

Please enter valid string (only alphabets and numbers allowed).

Create

[Back to List](#)

These shows all the error message when a user does not input the appropriate characters.

- **Sanitization**

Any input that the user enters will immediately be put through a sanitization process. This process transforms the data that the user inputs. It substitutes certain character that can be used in SQL Injection or Cross-Site Scripting such as <, >, @, !, =, ' , ' , &. These special characters are most likely to be sanitized and replaced with different symbols that are a common and safer alternative, which are not used in an SQL Injection attack.

Edit

Customer

Username

Name

Email

Address

Age

PhoneNum

Comment

[Back to List](#)

```
<tbody>
  <tr>
    <td>
      Terius2
    </td>
    <td>
      Ezra
    </td>
    <td>
      <script>alert('hi')</script>
    </td>
    <td>
      Jurong
    </td>
    <td>
      Jurong
    </td>
    <td>
      86983362
    </td>
    <td>
      This is Ezra Ho of CSF03
    </td>
  </tr>
</tbody>
```

Index

[Create New](#)

Username	Name	Email	Address	Age	PhoneNum	Comment	
Terius2	Ezra		Jurong	Jurong	86983362	This is Ezra Ho of CSF03	Edit Details Delete

- **Role-based Authorization**

Role-Based Access is essential to the website application. This is to ensure only certain roles can access certain pages and functions according to their authorized levels. These are all subjected to the access that is given by the owner to each role. Guests are only allowed to view the products and review section while customers can create, delete, and edit a review about a product they have purchased. This difference in authorization is due to the given access each role has been given. This highly mitigates any attacks on the database as only the owner has access to the database. Thus, any direct attack is highly unlikely, and this greatly reduces the possibility of an attack against the database.



The page is an example of an Access Denied page where the owner is unable to edit the product that has been listed by the seller.

1.8 Secure Software Testing

1.8.1 Test Strategy

After implementing the codes of several different functions, it is important to check the compatibility of the codes working together to achieve what the codes are meant to do. The testing strategy is used to help us test out the codes and ensure that they are properly coded and there are no small flaws which can cause huge security issues. Although the codes are tested through and through, there are many ways to strategize a test that will only test out specific parts of the code, whether it is to see the behaviors or check for logic bombs. The testing strategy employed is a combination of Black Box testing, White Box testing, Penetration testing, Vulnerability testing and Fuzzing.

- Black Box Testing

Black Box testing is a testing strategy that does not allow the tester to have any knowledge of the internal workings of the software presented. No inner workings of the software are revealed to the tester as this is to test the resiliency of the software, determining the outputs to the inputs of the tester as a form of behavioral analysis. This method can be used to identify and address security vulnerabilities proactively such that the risk of being attacked or hacked are minimized.

- White Box Testing

A White Box testing method is the opposite of black box testing. As the name suggest, it is where testers have full knowledge of the inner workings of the software designs and implementations. This is more commonly used to test use and misuse cases of the software. White box testing requires direct access to the source code to work, making it possible to detect embedded code issues such as spyware, backdoors, or logic bombs. The inputs are structurally analyzed to ensure that the code implementations follow a certain order and design specification.

- Vulnerability Testing

This test method is performed to detect and identify security flaws and weakness in the software such as trojans. It is critical part to ensure the software is not attacked by hackers, hence this test must be done periodically to provide updates and reports of the status of the software such that it can be more secure and up to date to protect itself. Most vulnerability scanners use a pattern matching, like signature-based IDS, against a database of vulnerabilities to detect and identify matching vulnerabilities. A flaw about signature-based vulnerability scanning is that the scanners are not able to detect new up and coming threats since it is not part of the vulnerability database list to be scanned for.

- **Penetration Testing**

Penetration testing, or Pen-testing, is commonly used to test input validation and sanitization of the software. This testing methodology will imitate an attacker that targets and exploits the software. This will scan for vulnerabilities that are currently available which may be able to compromise the security of the software. Attacks may include Brute Force using SQL Injection or Cross-Site Scripting attacks. When a single vulnerability is found, it will immediately be exploited. A report is then written up about the vulnerabilities found, and the code is restored to its original state.

- **Fuzzing**

Also known as Fault Injection Testing, a type of brute force software testing is used, where faults are injected into the software with the intention to observe the behavior of the software. The faults used can be SQL Injection statements or Cross-Site Scripting statements. The effectiveness of the validation and sanitization can also be tested along with this testing methodology.

1.8.2 Testing Plan

Our testing plan for the database consists of Functional Testing, Logic Testing, Input Validation Test, Injection Flaws Control, Scripting Attack Controls, Non-Repudiation Controls and Spoofing Controls.

- **Unit Testing**

- Unit testing consists of testing each individual unit of code to ensure that there are no software issues detected in each unit of code. This test can be conducted individually for each of us as we implement different parts of the code. The purpose of this is to validate each unit of the software, ensuring that it is performing as designed. Unit testing usually has a few inputs to be tested and results in a single output. Unit testing is the first step to checking and ensuring that the software will not give any errors

as these are small parts of the code and will be combined with other parts in integration testing to test for more software errors.

- Functional Testing
 - Functional test is performed to primarily test the functionality and reliability of the software. When we run to test the code, we are trying to see and review if the software is performing according to the expectations.
- Integration Testing
 - Integration testing is performed to ensure that the code is functional and reliable, secure and recoverable. This is used by this group when we push our codes into the repository at Bitbucket. The individual units of codes will be tested and aggregated to help validate and determine the functionality of the software. This is helpful to identify errors and issues when the units of codes are combined and together. While the units of code may pass unit testing, some will fail integration testing due to certain issues. Hence, integration testing is crucial to ensuring the security and usability of software.
- Logic Testing
 - This test validates the accuracy of the software processing logic. Validating the implementation details of the codes is critically important for the functionality and logic. Hence, input validation is put in place at many if not all of the inputs of the software. Predicate is also being tested, which is something that is affirmed or denied of the subject in a proposition in logic. Blind SQL Injection is also one of the examples of logic testing as well. All these tests can find issues and flaws for the security of the database and should be as thorough as possible to ensure that the database is well guarded and secure.

- Input Validation Test

- Special characters such as “!”, “@”, “#”, “\$”, “%”, “^”, “&”, “*”, “(”, “)”, “<”, “>”, “?”, “/”, “\”, “|”, “[”, “]”, “{”, “}”, “:”, “;”, “,”, “.”, “~”, “-”, “_”, “+”, “=” are generally not allowed in all inputs unless specifically allowed (e.g. address input allows “#” alongside “.”, as some users would like to include the specific house number in their address details. A full stop is allowed since short forms such as street becomes “St.”, thus requires a full stop.)
- When a user inputs an invalid input, it will display an error message beneath the input box, but will show a specific message on what to input, and not show the entire regular expression so as to allow the attacker to brute force in an attempt to break the site.

Validate for	Regular Expression	Description	Example
Toy_Name	<code>^[a-zA-Z0-9_-'&]+([a-zA-Z0-9_-'&]+)*\$</code>	Only allows alphabets and numbers. Spaces are allowed too. No special characters are allowed except “_”, “-”, “'”, “&”.	Lego City Fire Station 60215 Super Wings Transforming Build-It Jett
Age	<code>^[a-zA-Z0-9_+]+([a-zA-Z0-9_+]+)*\$</code>	Only allows alphabets and numbers. Spaces are allowed too. No special characters are allowed except “_” and “+”.	5+ and above 10 and above
Price	<code>@^[0-9]{0,2}((.[0-9]{0,2}))*\$</code>	There must be a number in the dollar place and two more in the cents place. Only numbers are allowed.	99.90 123.30
Category	<code>^[a-zA-Z0-9_]+([a-zA-Z0-9_]+)*\$</code>	All alphabets and numbers, special characters are not allowed except “_”.	Lego, Technic, Cars, Vehicle
Rating	<code>@^[A-Z]+[a-zA-Z0-9_.,']*\$</code>	It must start with an alphabet and allows numbers. Special characters are not allowed except “_”, “.”, “'”, “,”.	This item, the car, is so good. It's a dream come true.

- Injection Flaws Control Test

- Injection Flaws enable attackers to input malicious code through an application to another system. It can be though the use of external programs via shell commands and backend databases using SQL Injection.

- Sources of input and the events where the software will be connected to the backend store or command environment. The sources can range from authentication forms, search input fields, or hidden fields in the webpages. When sources are determined, input validation tests are performed to ensure software is not susceptible to injection attacks.
- Scripting Attacks Controls Test
 - The lack of output sanitization will cause scripting attacks. This test is performed to validate controls which mitigates scripting attacks. It ensures that the output is sanitized by escaping or encoding the input before it is sent to the client. Another one of the many tests to ensure is that scripts are unable to be injected into the input fields.
 - One way we tested on our own page was to log in as a seller and attempt to create a listing. Instead of listing an actual product, attempt to inject scripts into the webpage. The page will deny any scripting performed. If any scripts passed, check the source code to find that the scripts have all been sanitized before it was submitted into the database. The same can be performed for customer and owner input fields.
- Non-Repudiation Controls Test
 - Audit trails are necessary to track which user did which action to verify and provide proof of non-repudiation. Tests are to assure that an attacker is unable to exploit the audit page should any code or unauthorized action is performed. Confidentiality of the audit page and the codes on which it is performed should remain hidden from the customers and only accessible to the owner.

- Spoofing Controls Test
 - The test to determine spoofability of the user and certificate alongside verifying the presence of transport layer security can ensure secure communication and protection against Man-In-the-Middle (MITM) attacks. Setting a short cookie expiration in addition to verifying that authentication cookies are encrypted are critical to the security of the website.

1.8.3 Test Cases

Number	Test Scenario	Test Case	Test Condition	Test Steps	Test Data	Expected Results	Actual Result	Does Actual Result match with Expected Results?
1	Create Customer	Fill up customer details to create a customer.	The user is logged in as an administrator, having administrative rights.	1. Logged in as Admin. 2. Press the create button.	User is valid.	The admin will be allowed to create the customer.	The admin can create the customer.	The actual result does match the expected result.
2	Create Customer	Fill up customer details to create a customer	The user is logged in as a customer, having limited rights.	1. Logged in as a Customer. 2. Press the create button.	User is invalid.	The customer will not be allowed to create the customer.	The customer is not able to create customer.	The actual result matches the expected result.
3	Create Toys	Fill customer details to create a customer.	The user is logged in as an administrator, having administrative rights.	1. Logged in as an Admin. 2. Press the create button.	User is invalid.	The admin will not be allowed to create a toy listing.	The admin is not able to create the customer.	The actual results match the expected result.
4	Create Toys	Fill customer details to create a customer.	The user is logged in as a seller, having seller rights.	1. Logged in as a Seller. 2. Press the create button.	User is valid.	The seller will be allowed to create a toy listing.	The seller can create the customer.	The actual results match the expected result.
5	Create Audit Log	Fill audit details to create an audit log.	The user is logged in as an admin, having admin rights.	1. Logged in as an admin. 2. Press the create button.	User is valid.	The admin will not be allowed to create an audit log.	The admin can create the audit log.	The actual results match the expected result.
6	Create Audit Log	Fill audit details to create an audit log.	The user is logged in as a customer, having customer rights.	1. Logged in as a customer. 2. Press the create button.	User is invalid.	The customer will not be allowed to create an audit log.	The customer can create the audit log.	The actual results match the expected result.
7	Check for input validation on Category field	Input an SQL Injection Statement	User is logged in as a seller and have seller rights.	1. Goes to Toy Create page 2. Key in toy information and purposely inject SQL statements into the input field e.g. <code><h1>Test</h1></code>	User is valid.	The Toy will not be created as an SQL injection warning will appear beneath the input field and user is unable to submit.	"Please enter valid string (only alphabets and numbers allowed)."	The actual results match the expected result.
8	Check for sanitization of inputs in email field in Create Customer Page.	Input an SQL Injection statement.	User is logged in as an administrator and have administrator rights.	1. Goes to Create Customer page 2. Attempt to create or change category field by injecting it with SQL statement e.g. <code><h1>Test</h1></code>	User is valid	The customer will be created. However, the email field that was injected with SQL statements will be blank.	The customer will be created, and email field is left blank. Sanitization has happened.	The actual results match the expected results.
9	Check for input validation on Category field	Input a Cross-Site Scripting statement	User is logged in as a seller and have seller rights.	3. Goes to Toy Create page 3. Key in toy information and purposely inject SQL statements into the input field e.g. <code><script>alert("Hi")</script></code>	User is valid.	The Toy will not be created as an SQL injection warning will appear beneath the input field and user is unable to submit.	"Please enter valid string (only alphabets and numbers allowed)."	The actual results match the expected result.

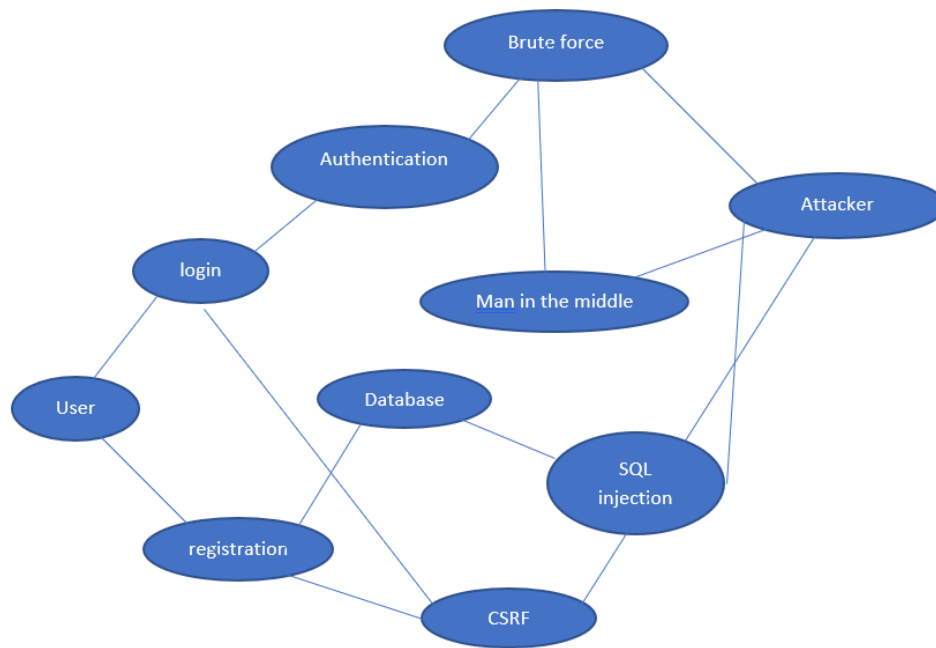
1.9 Security Conclusion of the Feature

In conclusion, the CRUD Database feature is critical to the entire process of the website application to be fully functional as it is the mastermind of the entire operation. This must remain secure and highly confidential in the operation as to not allow attackers easy access to the database. With the proper authorization for each of the users in place, the proper tier of accessibility and rights are allocated to the user. Having the users authorized would give the owner an easier time to maintain the website application against unwanted attacks. Thus, these two features of Authorization and the CRUD Database are both secure and have been properly tested and would keep the website secure.

Feature 2: Registration & Login

2.1 Secure Software Requirements

2.1.1 Use Case and Misuse Case Modeling (done)



2.1.2 Data Classifications (done)

Customer data will be stored in the database. The data stored has to be classified as it contains certain confidential information such as email address and perhaps the payment details. The data stored in the database requires certain traits such as Confidentiality, Integrity and Availability (CIA) as they are confidential and critical to the company. Username need not be labeled as high confidentiality, but information such as credit card details is of high confidentiality, hence is hashed using SHA 256 in order for the data to remain protected. This will ensure that no one is able to take advantage of confidential information from the database, which can earn more trust of the customers.

2.1.3 Data Labelling (done)

Attribute		Confidentiality	Integrity	Availability
ID	A unique ID assigned to every customer in the database.	Restricted	High	High
Username	A username to identify the user.	Public	Low	Medium
Normalized Username	Contains the uppercase value of the username.	Public	Low	Medium
Email	Email address of user.	Private	High	Medium
Normalized Email	The uppercase email address of the user.	Private	High	Medium
Email Confirmed	This is a true or false result, showing if the email address used by the user is confirmed or not.	Restricted	High	Medium
Password Hash	Local phone number of users	Restricted	High	High
Security Stamp	To track changes made to a user account e.g. changing the password.	Restricted	High	High

Concurrency Stamp	To prevent multiple parties from being able to modify the same data at the same time within the database.	Restricted	High	High
Phone Number	Phone number of customers.	Private	Medium	Low
Phone Number Confirmed	Is a Boolean output which shows true when phone number is confirmed via OTP (One-time password)	Restricted	Medium	Low
Two Factor Enabled	Shows a Boolean output where it displays true if the customer has 2-factor authentication enabled for the account.	Restricted	High	High
Lockout End	Displays the date and time at which customer can access the account again.	Restricted	High	High
Lockout Enabled	Displays a Boolean expression of true if account is locked out, and false if account is not locked out.	Restricted	High	Medium

Access Failed Count	Counts the number of times the user has failed log into the account.	Restricted	High	Medium
Full Name	This displays the full name of the customer.	Public	Low	Medium
Birth Date	The birth date of the customer is shown here.	Public	Low	Low
Age	The age of the customer is displayed.	Public	Low	Low

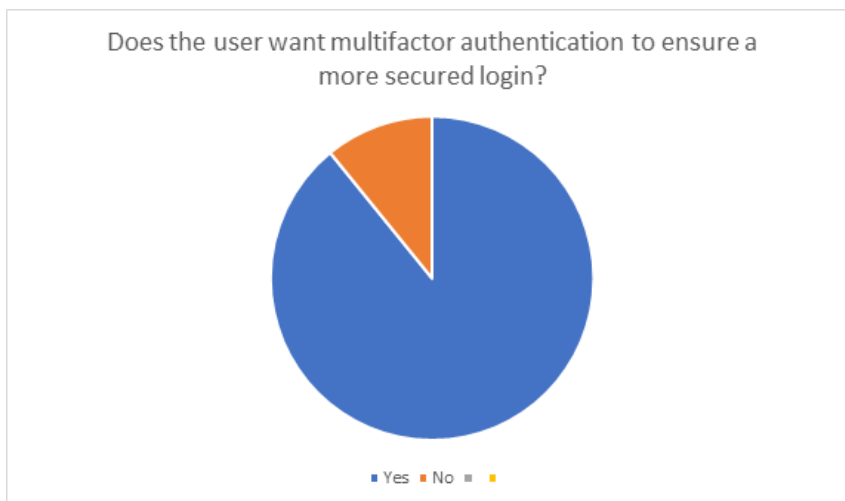
2.1.4 Subject/Object Matrix

Attribute	Create (C), Read (R), Update (U), Delete (D), None (-)
UserID	-
Username	CRU
PasswordHash	-
Name	CRU
DOB	CRU
Email	CRU
Address	CRU
PhoneNum	CRUD

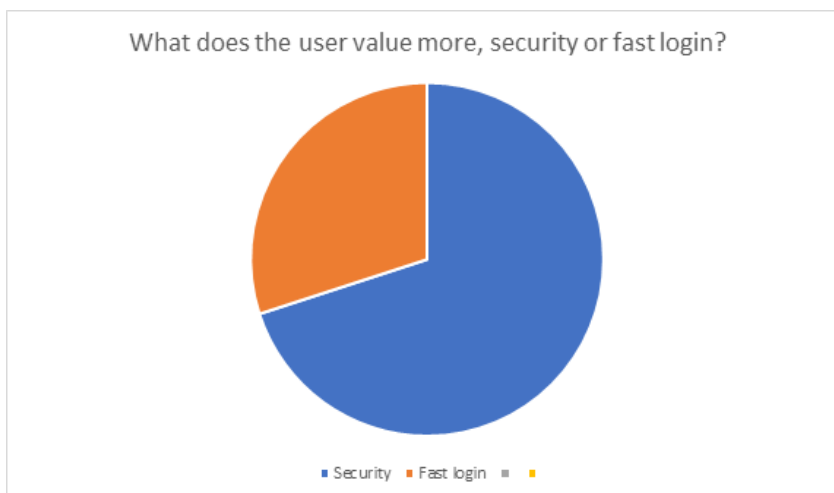
2.1.5 Surveys (need 1 more question then finish alrd)

- What is the purpose for registering an account on the ecommerce website.
 - It is to allow customer to have their account details saved which ease the process of checking out. Furthermore, they could leave feedback about a certain product and help other interested buyers make a purchase.

- What information are the websites and account obtaining from you?
 - The information required will be personal information about yourself e.g. email, password, and name. Certain process might require information such as address, credit card details and phone number.
- What information should be classified and confidential?
 - All information should be confidential as it is sensitive information except the username of the account.
- Does the user want multifactor authentication to ensure a more secured login?



- What does the user value more, security or fast login?



2.1.6 Relevant Security Concepts

- **CIA**

- Confidentiality

- Data is being stored in database servers must be encrypted. Otherwise, attackers can more easily obtain data in the database.
 - Data that is transmitted from user to web application to database and vice-versa needs to be secured with SSL (Secure Sockets Layer) to reduce the likelihood of man in the middle attack.
 - Only the owner, who is authenticated and verified, can have access to the database.

- Integrity

- Integrity is ensuring that the data that was added into the database has been authorized and no unauthorized modifications, creation or deletion of data was made.
 - In an event of SQL injection in the database, the integrity of the data might be questionable.
 - Inputs should be sanitized and validated before being processed by the servers and added into the database.

- Authorization

- Each user, when registered for an account, is assigned an authority that prevents users from accessing data they are not allowed to access.
 - The owner has total control over the database and is the only one who has exclusive access to the database.

- **AAA**

- Accountability

- Any action performed by the customer and seller will be tracked and audited to provide a list to be used as evidence for non-repudiation. The person who performed the action and the action committed will be audited.

- The actions and user data recorded will be stored into the database of the server and audit logs.
- Authentication
 - Users must go through the email confirmation process before they are able to access the home page. This ensures that their email address is authentic and is accessible to them in case of password recovery.
 - Users can enable 2 factor authentication and one-time password (OTP) if they wish to enhance their security further.
- Availability
 - By increasing the number of servers, there will be a lesser chance that there will be a failure which results in a loss of revenue for the owner.
 - This ensures that customers and guests are able to browse and log into the website at any time of the day, any day of the week.
- Session Management
 - There should be randomly generated session ID which are then encrypted via SSL.
 - Cookies should expire after a certain period to prevent attackers from using the cookie session.
- Error Pages
 - When a user just registered for an account, the user must first verify their email address before they are allowed entry into the webpage. Otherwise, an error will occur, and the image shown below will be displayed.

Register confirmation

Please check your email to confirm your account.

2.2 Secure Software Design

2.2.1 Feature related security design

By implementing authentication features such as 2-factor authentication, email confirmation, external login via Google, Facebook and Microsoft as well as SMS Confirmation (not in code), we are able to ensure that the authenticity of the customer is genuine upon registration. This is crucial to the process of securing the website application as attackers would be able to create multiple dummy accounts and attack the website application. Having these measurements in place mitigates the risks of attacks.

- Email Confirmation
 - How does it work?
 - For any customer to have access to their account after registration, they have to verify their email they used to register the account by clicking the verify button in their email.
 - This is to ensure the end customer is authenticated to prevent any impersonation and misuse of personal credentials.
- SMS Confirmation
 - How does it work?
 - For any payment to be made, a one-time password (OTP) confirmation will be sent out and it needs to be verified before any purchase could be made. This is to ensure the credentials of the users will not be misused if any attacker manage to gain access of the account.
 - With this added security feature the end user can be authenticated to ensure that the person who registered for the account is genuine and not for malicious purposes.
- 2 Factor Authentication
 - How does it work?

- Users are only able to enable 2-factor authentication once they have confirmed their email address. This will allow for more security for ensure that the account of the customer is only used by the customer and not anyone else.
- This is not a compulsory feature for customers to implement as it might complicate some customers.
- But upon activation, customers will have to scan the QR code that is shown on the screen when they login to allow the login to authenticate and verify that it is indeed the customer and not a hacker.

2.2.2 Attack Surface Evaluation

There are many attack surfaces that a hacker can exploit

- Cross Site Scripting (XSS):
 - Special symbols and characters can easily be used to gain unauthorized access to the database of the organization.
 - One way to mitigate cross site scripting is the sanitize and validate the inputs when guests and customers are entering it in the input field.
 - Another way to mitigate is to enforce strict rules on characters used and encoding.

```
[Required]
[EmailAddress]
[Display(Name = "Email")]
7 references
public string Email { get; set; }

[RegularExpression("^[a-zA-Z0-9_]+([a-zA-Z0-9_]+)*$", ErrorMessage = "Please enter valid string (only alphabets and numbers allowed).")]
[Required]
[StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 6)]
[DataType(DataType.Password)]
[Display(Name = "Password")]
4 references
public string Password { get; set; }

[RegularExpression("^[a-zA-Z0-9_]+([a-zA-Z0-9_]+)*$", ErrorMessage = "Please enter valid string (only alphabets and numbers allowed).")]
[DataType(DataType.Password)]
[Display(Name = "Confirm password")]
[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
3 references
public string ConfirmPassword { get; set; }

[RegularExpression("^[a-zA-Z ]*$", ErrorMessage = "Please enter a valid name (only alphabets).")]
[Display(Name = "Full Name")]
4 references
public string FullName { get; set; }
```

- SQL Injection
 - This uses SQL statements to carry out a specific command on the database which is not intended by the owner of the website.
 - One way to mitigate an SQL Injection attack is to use input validation to filter out special characters. These characters can be symbols, characters or even punctuation.
 - Another way to mitigate is using error pages to redirect errors from exposing the issue caused by the database. This might reveal vulnerable information to the hackers.

- Brute force
 - Brute force is when an attacker constantly and repeatedly tries to use different passwords to gain access into an account.
 - One way to mitigate this is by having a stronger password, which can consist of alphanumeric characters, which are alphabets and numbers.
 - Another way is to impose timeouts for a certain amount of time when a specified number of unsuccessful attempts has been achieved.

Surface	Is it exploitable by the hacker?	Mitigation to an attack by hacker.
Registration Page	Yes. The registration page has input fields which allow hackers to input malicious codes.	Input validation and restriction on certain characters and symbols in the input field prevents SQL Injection and Cross-site scripting.
Login Page	Yes. The login page has input fields which allows hackers to input malicious codes.	Implementing input validation, restricting certain characters and symbols as well as having input sanitization on the input field prevents SQL Injection and Cross-site scripting.

2.3 Secure Software Coding

2.3.1 Feature related Defensive Coding Practices

- Passwords are well constructed

Team2_SSD_Assignment Home Product Register Login

Register

Create a new account. Use another service to register.


• The Password field is required.

Email
admin1@gmail.com

Password
The Password field is required.

Confirm password

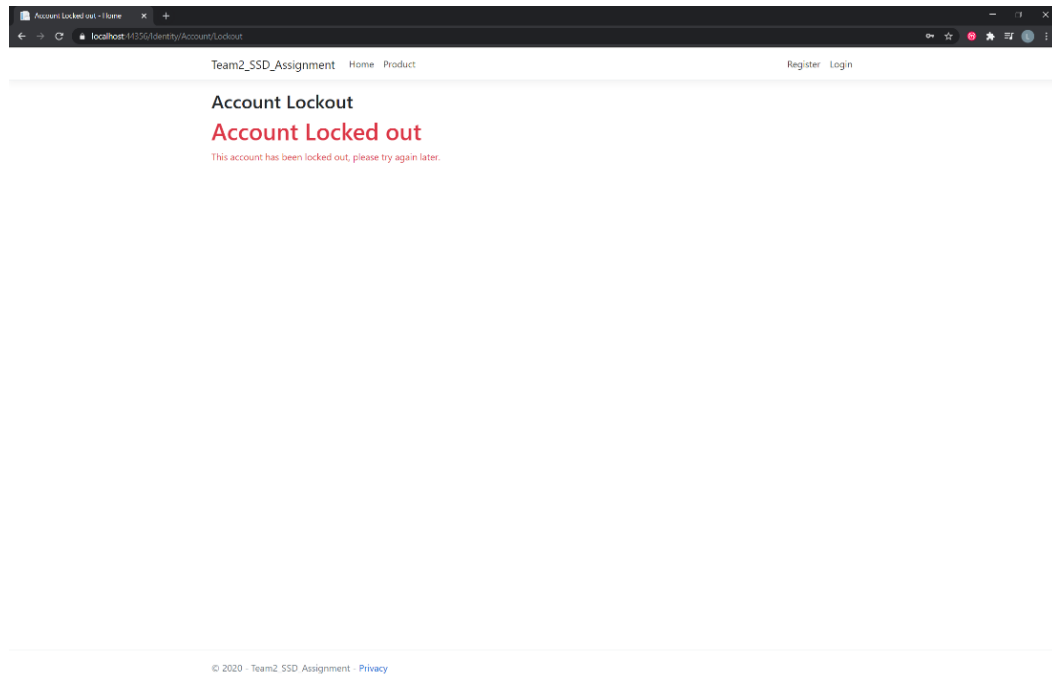
Full Name
hi

☒ I'm not a robot 

Register

-

While registering an account, the password needs to have a mix of numbers and alphabets, both in upper and lower case as well as special character to ensure a well-constructed password with little possibly of a password being guessed by any attacker easily.



With the addition of this feature to the login page, it will prevent attackers from using brute force attacks as there will be a timeout of 10 minutes after 5 unsuccessful attempts. With the addition of this it will hinder the effectiveness of a brute force attack.

2.4 Secure Software Testing

2.4.1 Test Strategy

After implementing all our security features, it is important we test all our features that were implemented to ensure they all in working condition to prevent any misuse of feature or flaws that could be exploited. Hence, we will put our feature through a series of different test to try our best to exploit any flaws to the best of our abilities.

- Black box testing

This method of testing is the closest to what an attacker will do. They have no access to any internal information and isn't granted internal access hence it is the job of the tester to perform all reconnaissance to obtain all the information and knowledge he needs to proceed his attacks. Hence, we felt that this method of attack would be the most realistic in the real-world context.

- White box testing

This method of testing allows the tester to have complete open access to our application and system. This allow him to view the source code and be granted a high level of privilege accounts to the network. This allows the tester to identify potential weakness in various areas such as logical vulnerabilities, poorly coded web development, lack of defensive features/measures and potential security exposures

2.4.2 Test Plan (not done)

Going into this we have decided that the main concern would be the input validation of our website as it was the first entry point for anyone to access. Though we have coded our page to sanitize all the input, we will still put it to test to ensure all the input are validated properly and it doesn't trigger any commands that aren't supposed to be.

- Input validation test
 - To perform this test, we will try to put the special characters like "!", "@", "#", "\$", "%", "^", "&", "*", "(", ")", "<", ">", "?", "/", "\\", "|", "[", "]", "{", "}", ".", ",", "~", "-", "_", "+", "=" and it should be unallowed to be input.
 - When the user input an invalid character, the display error message will be shown and display they types of expressions that could be input
- Injection test
 - To perform this test, we will input special scripting lines and test if it triggers any database information leak.
 - When the user input a special line, an error message should show up straight away and request the user to input again
- Functional test
 - This test is needed to ensure the coded features works properly without any errors
 - During the testing process we will test the login functions and input validations.

2.4.3 Test Cases

Test case scenario	Data input	Expected results	Actual results
Input validation	Email: bobbyLam783#@& Password: password321	Email not accepted; wrong format Password not accepted; too weak	Incorrect format of email and password
Input validation	Email: khoocheng89@yahoo.com Password:uy8	Password not accepted; length too short	Password not accepted
SQL injection scripting attacks	<script>alert("xss"); </script>	All input fields should not accept this data	Account not registered and script not loaded into database due to incorrect input format
Error and exception handling	Blank input fields	Unable to register an account and no critical failure points and information will be shown	Unable to register an account as the web application will show the incorrect input fields that needs to be changed

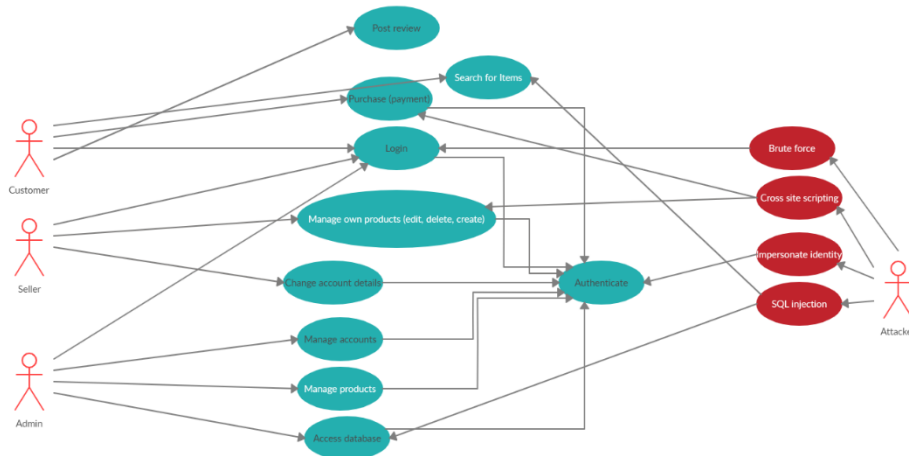
2.5 Security Conclusion of the feature

Registration and login were coded and tested to the best of our abilities to fulfil the CIA security traits. We have aimed to code it to prevent common attacks such as SQL injections, brute force and cross site scripting attacks. With login and registration page being the first entry point for any attacker it should be coded with more security features to ensure the reliability of the page. We have tried and tested it vigorously to ensure it have no loop holes that could be exploited.

Feature 3: Authentication

3.1 Secure Software Requirement

3.1.1 Use case and Misuse Case Modeling



3.1.2 Data Labelling and classification

The data of the users that is used for authentication will be stored in the database and will be protected with appropriate encryption. For example, password will be stored hashed as to ensure it does not show the plain password and even when the attacker hacked into the database, the attacker cannot see the password in plaintext. And as for data for 2FA, it is not stored in the database and there is a time expiry for those data. Thus, those are not needed to be store in the database. However, it is crucial that that data must not be leaked and the third party for providing the 2FA services will be responsible as the transportation of those data must not be leak and must be sent to the user in a safe and secure way.

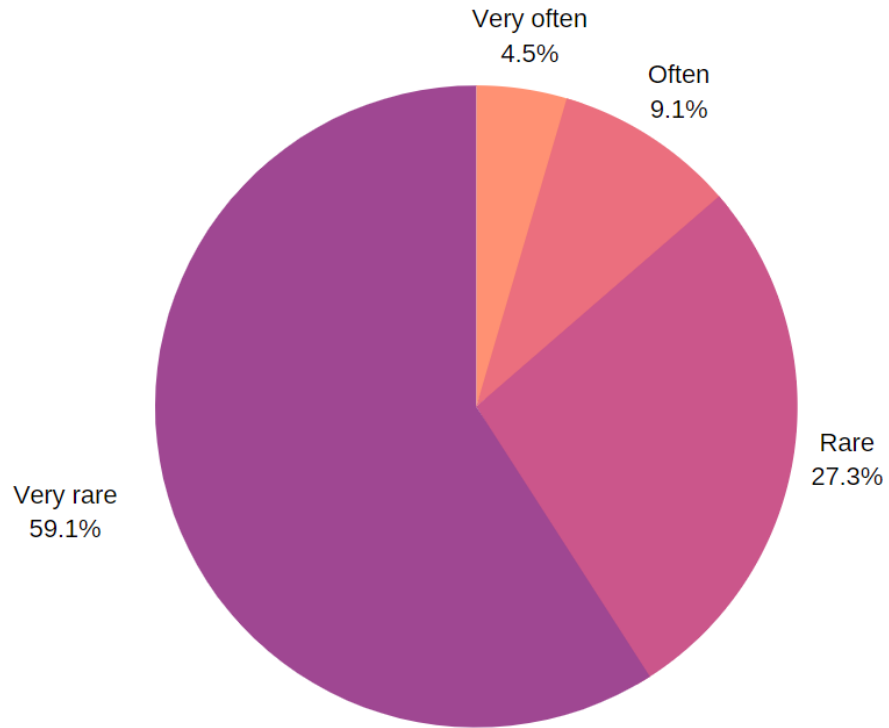
Attribute		Confidentiality	Integrity	Availability
ID	A unique ID assigned to every customer in the database.	Restricted	High	High
Username	A username to identify the user.	Public	Low	Medium
Email	Email address of user.	Private	High	High
Password	Local phone number of users	Restricted	High	High

3.1.3 Subject/Object matrix

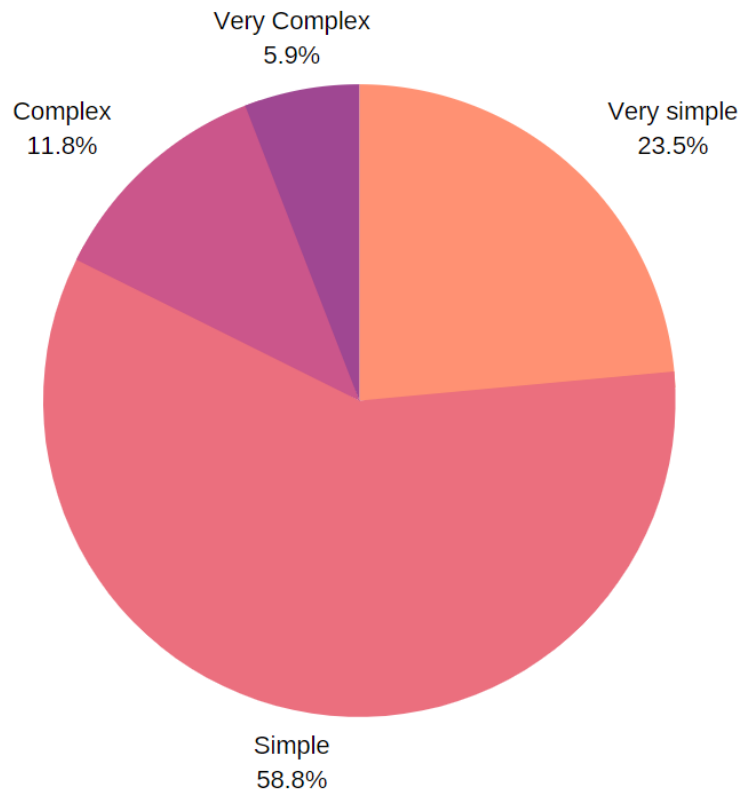
Attribute	Create (C), Read (R), Update (U), Delete (D), None (-)
UserID	-
Username	(user)CRU, (owner)CR, (seller) CRU
PasswordHash	-
Name	(user) CRU, (owner)CR, (seller)CRU
DOB	(user) CR, (owner) CR, (seller)CR
Email	(user) CRU, (owner)CR, (seller)CRU
Address	(user) CRU, (owner) CR, (seller)CRU
PhoneNum	(user) CRUD, (owner) CR, (seller)-CRUD

3.1.4 Surveys

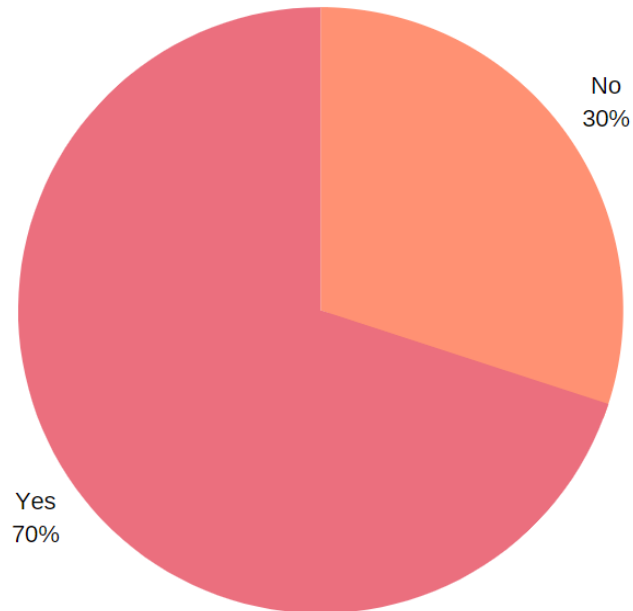
- A. The main reason for authenticating any user before accessing their account.
 - I. This is to prevent hackers from misusing confidential credentials saved in the user account which could allow them to be impersonated.
- b. What information need to be authenticated by the website?
 - I. The user's personal credentials must be authenticated before being saved to the account for example email account, mobile numbers and credit card credentials.
- c. How to ensure every user will be authenticated to prevent any impersonation.
 - I. Ensure there is multi factor authentication every time a user logs in to their account.
- d. How often do you change your password?



e. How complex do you set your password?



f. Would you use 2FA?



3.1.5 Relevant Security Concept and Management

3.1.5.1 Confidentiality

Data being transferred from user to web application to database should be confidential and secured

3.1.5.2 Integrity

Ensure the reliability of the data and to prevent unauthorized modification to data

3.1.5.3 Availability

Ensure that the business continue to prevent loss of revenues for the organization.

3.2 Secure Software Design

3.2.1 Feature Related security design

3.2.1.1 Federation

Allows an individual to log into the site and access the products at another affiliated site without having to log in each time or re-establish their identity.

3.2.1.2 Multi-Factor Authentication

Validating and verifying using 2FA via QR code. The site will display a QR code for the user, and the user will have to scan it and then they are validated and verified.

3.2.1.3 Single Sign On (SSO)

Once the user is logged in, their verified credentials are passed on to other systems when needed. Federation is an extension of SSO into other enterprise. It simplifies credential management and improves user experiences and performance because their credential is verified only once.

3.2.1.4 Email Confirmation

Users will have to verify and confirm their email account before they can login into the site. This is to ensure that the user from impersonation and misuse of personal credentials.

3.2.1.5 ReCaptcha

This is used so that the user is authenticated as a human and not a bot, this is so that bots cannot spam attack by creating a huge amount of account and then causing the site to lag and denying services

3.2.1.6 SMS Confirmation

For any payment to be made, a one-time password (OTP) confirmation will be sent out and it needs to be verified before any purchase could be made. This is to ensure the credentials of the users will not be misused if any attacker manage to gain access of the account. With this added security feature the end user can be authenticated to ensure that the person who registered for the account is genuine and not for malicious purposes.

3.2.2 Threat modeling

STRIDE	Security Property	Identified threat
Spoofing	Authentication	<ul style="list-style-type: none"> • Brute Force Authentication • Cookie Replay • Session Hijacking • CSRF
Tampering	Integrity	<ul style="list-style-type: none"> • Cross-Site Scripting (XSS) • SQL Injection
Repudiation	Non-repudiation	<ul style="list-style-type: none"> • Insecure Backup
Information Disclosure	Confidentiality	<ul style="list-style-type: none"> • Eavesdropping Verbose Exception • Output Caching • Path Traversal
Denial of Service	Availability	<ul style="list-style-type: none"> • Website Defacement
Elevation of Privilege	Authorization	<ul style="list-style-type: none"> • Logic Flaw

Threat	D	R	E	A	DI	Total	Result
SQL Injection	3	3	2	3	2	13	High
Cross Site Scripting	3	3	3	3	3	15	High
Cookie Replay	3	2	2	2	2	11	Medium
Session Hijacking	2	2	2	1	3	10	Medium
Cross Site Request Forgery	3	1	1	1	1	7	Low
Verbose Exception	2	1	2	3	1	9	Medium
Brute Force	2	1	1	3	2	9	Medium
Eavesdropping	2	1	1	2	2	8	Medium
Insecure Backup	2	1	1	2	2	8	Medium
Audit Log Deletion	1	1	1	1	3	7	Low
Output Caching	3	3	2	3	3	14	High
Website Defacement	3	2	1	3	3	12	High
Logic Flaws	3	1	1	2	1	8	Medium

Threat	Probability of Occurrence (P)			Business Impact		P	I	Risk
	R	E	DI	D	A	(R + E + DI)	(D + A)	P * I
SQL Injection	3	2	2	3	3	7	6	42
Cross-Site Scripting	3	3	3	3	3	9	6	54
Cookie Replay	2	2	2	3	2	6	5	30
Session Hijacking	2	2	3	2	1	7	3	21
CSRF	1	1	1	3	1	3	4	12
Verbose Exception	1	2	1	2	3	4	5	20
Brute Force	1	1	2	2	3	4	5	20
Eavesdropping	1	2	2	2	2	5	4	20
Insecure Backup	1	1	2	1	2	4	3	12
Audit Log Deletion	1	1	3	1	1	5	2	10
Output Caching	3	2	3	3	3	8	6	48
Website Defacement	2	1	3	3	3	6	6	36
Logic Flaws	1	1	1	3	2	3	5	15

3.2.3 Attack surface evaluation

- Brute force attack

Attack can happen at the login page where the attack can make use of customized password dictionary to gain access to the account. The attacker and keep trying different password to gain authentication and gain access to the user account.

Mitigations:

- Implement a strong password policy
- Retain password history to prevent reuse of password
- Implement account block out when multiple incorrect passwords are used

- Man in the middle attack

Attack can happen at the login page when the authentic user is attempting to log in to their account and the attacker is placed between the client and server and the client is fooled into authenticating to the attacker.

Mitigations:

- Use encrypted communications
- Perform multi factor authentication

- Cross Site Scripting (XSS):

Special symbols and characters can easily be used to gain unauthorized access to the database of the organization. By using special symbols and characters, the attacker can write a code to attack the site

One way to mitigate cross site scripting is the sanitize and validate the inputs when guests and customers are entering it in the input field.

Another way to mitigate is to enforce strict rules on characters used and encoding.

Another way is through authentication, authenticating and verifying the user's token to prevent cross site scripting.

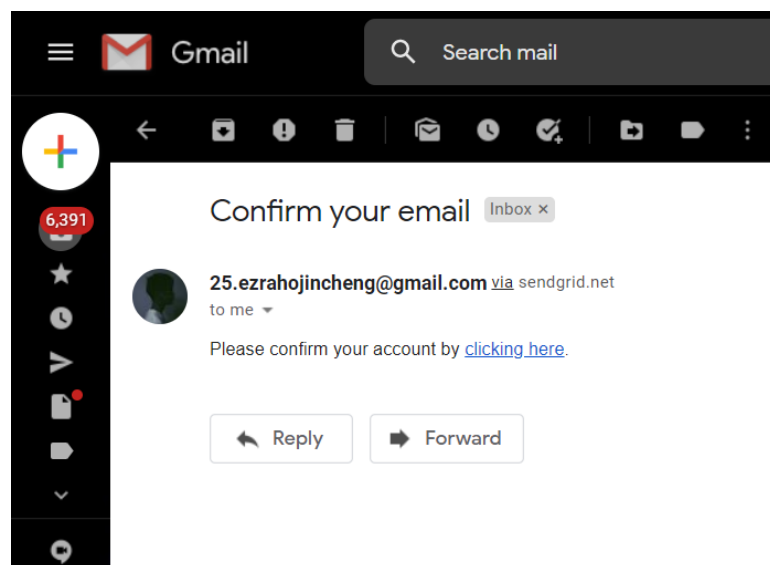
3.3 Secure software coding

3.3.1 Feature related defensive coding practices

To ensure we authenticate every user before they are allowed access to their account or make any purchase we have added multi-factor authentication. This is to prevent impersonation and misuse of any account. To maximize the security level, we have placed multifactor authentication at a few points such as the entry point, purchasing point and login.

1. Email confirmation

- A. Upon registering, the user must authenticate their email before having access to their account. An email will be sent to their email that they have chosen to register with, the user then needs to click it to authenticate it. With this feature being put into place it will increase the security level and authenticate every new user.



B. SMS confirmation

- Before being able to purchase any item with the saved credentials in the user account, they need to go through another round of multi factor authentication in the form of SMS confirmation to verify they are the owner of the account. This will prevent any misuse of credit card credentials.

C. 2FA

- If the user enables this feature, then every time when the user logs in, the user will be prompted to enter the 2FA code provided by the authentication app (google authentication). This will help the to verify the user and prevent impersonation.

3.3.2 Secure Software processes

3.3.2.1 Versioning

This ensure that all members on the team is working on the most up-to-date version of the code, granting rollback if necessary. This has a secondary function, which is the audit log, displaying the changes in code and the users who pushed the changes. We coded on our individual devices and used Bitbucket as a platform to compile and update our codes for others.

3.3.2.2 Static Code Analysis

This is the process of inspecting the code in Visual Studio 2019 to figure out which codes may have the potential to cause errors. When codes are committed and pushed, the codes that are pulled by the teammates might cause errors due to incompatibility issues. The errors are indicated by a red underline in Visual Studio.

3.3.2.3 Dynamic Code Analysis

Dynamic inspecting involves inspecting the code during execution to figure out if there are any errors present. One example of an error is “build error”, where why the project can compile, but has issues when it tries to run. This is critical to know the errors that are still present in the project, ensuring that

it can run as expected us users. The reason for these errors is perhaps certain parameters within the database not defined.

3.3.2.4 Code & Peer Review

Code is tested and reviewed by other members in the team to highlight any mistakes and to correct it before proceeding further. These are not to point out mistakes but learning points to improve the application when such a mistake occurs. Ultimately, it lowers the vulnerability and errors caused by the application when it is finished.

3.4 Secure software testing

3.4.1 Test strategy

After implementing all our security features, it is important we test all our features that were implemented to ensure they all in working condition to prevent any misuse of feature or flaws that could be exploited. Hence, we will put our feature through a series of different test to try our best to exploit any flaws to the best of our abilities

- Black box testing

This method of testing is the closest to what an attacker will do. They have no access to any internal information and isn't granted internal access hence it is the job of the tester to perform all reconnaissance to obtain all the information and knowledge he needs to proceed his attacks. Hence, we felt that this method of attack would be the most realistic in the real-world context.

- White box testing

This method of testing allows the tester to have complete open access to our application and system. This allow him to view the source code and be granted a high level of privilege accounts to the network. This allows the tester to identify potential weakness in various areas such as logical vulnerabilities, poorly coded web development, lack of defensive features/measures and potential security exposures

3.4.2 Test plan

The test plan was to test whether authentication works and test whether the site will be able to verify the user.

- Impersonation attack

To test this, we must give invalid tokens, and it is not supposed to give access to the user that gave the invalid tokens. For example, it is not supposed to let the user gain access when they provide a wrong 2FA code.

- Invalid Email

To test this, we must provide an invalid email and it is not supposed to let the invalid email to be authenticated and gain access to the account. In addition, we must verify that the email is valid, to do this, we can send an email to the user and the user must click the link in the mail to verify their email.

- Invalid password

To test this, we must provide a wrong password and the site is not supposed to allow the user to gain access when the password is invalid. For example, during log in, if the user provides the correct email, but the wrong password, they should not be allowed to have access to the account

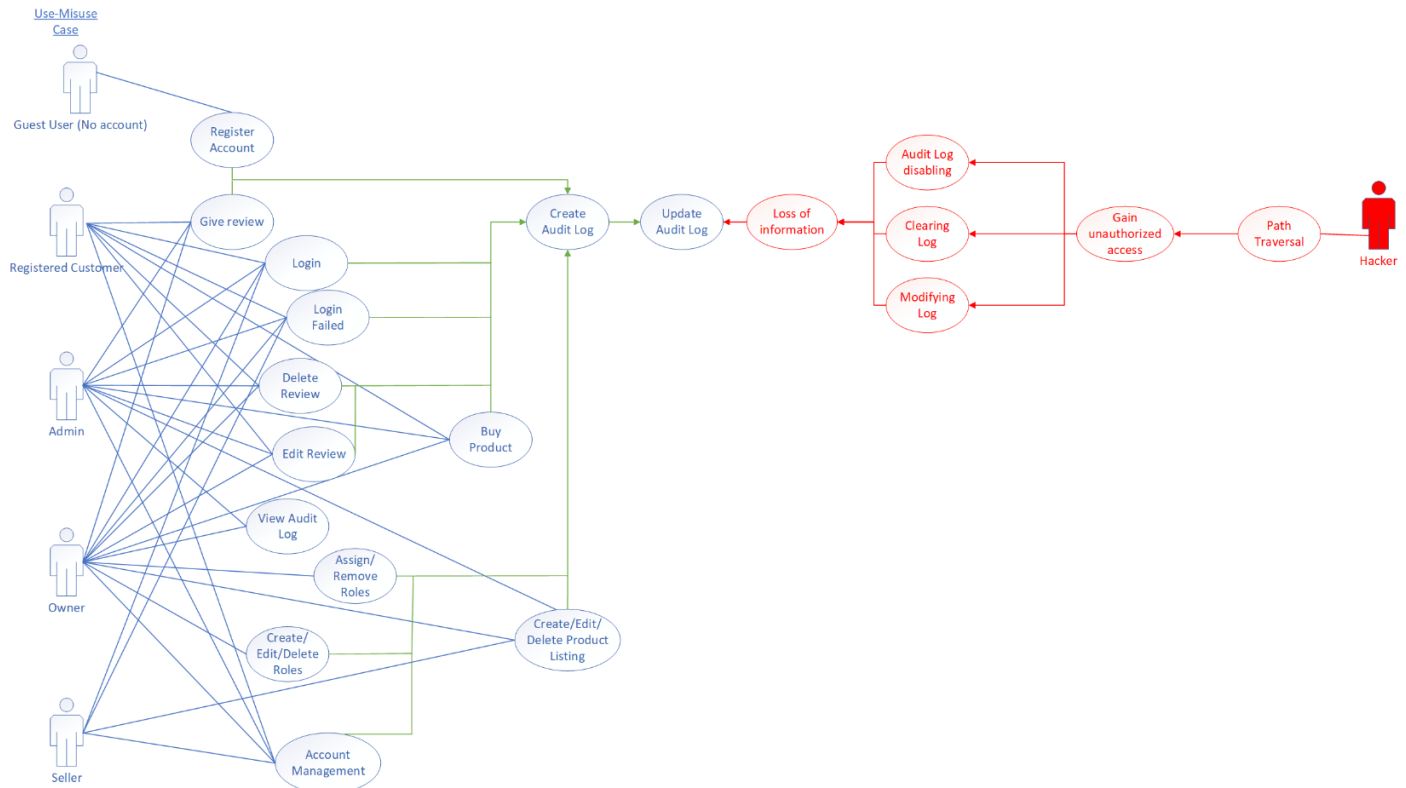
3.4.3 Test cases

Test case scenario	Expected results	Actual Result
Email-Confirmation	Email Confirmation sent to user and email is able to be confirmed	Email Confirmation sent to user and email is able to be confirmed
2FA	Can be scanned and verify the account	Worked as expected and was able to gain access to the account
SMS Confirmation	SMS is sent to user phone upon payment and payment can only be done after the OTP have been verified	SMS works perfectly fine, and payment can go through after OTP
Federation	To be able to sign in with external logins like Google and Facebook	Works fine and is able to log in with Google and Facebook
ReCaptcha	Can register only after the Captcha checkbox is checked	Can register when checkbox is checked

Feature 4: Audit

4.1 Secure Software Requirements

4.1.1 Use Case and Misuse Case Modeling



4.1.2.1 Data Labelling

The page will first display the Audit Log ID which is hidden, followed by the LogInID (user's assigned id: eg, john#1234) of the user who performed the action, the action type such as edit, create, delete, accessed. The productID of the change made, will be 0/NULL if account was created or logged in. And lastly, the date and time the action was performed.

Display Example:

Log ID [hidden]	Performed By	Audit Action	Listing ID	Role	Date/Time Stamp
2	0001	Edit	1	Seller	23-11-2020 11:59
1	0121	Create	0	Seller	22-10-2020 23:59

4.1.2.2 Data Classification

Data	Impact upon loss		
Audit Log data	Confidentiality	Integrity	Availability
Log ID	Low	High	Low
LogInID	Low	High	Low
Action	Low	High	Low
ProductID	Low	High	Low
DateTime	Low	High	Low

4.1.3 Subject/Object Matrix

Data	Roles				
	Guest	Customer	Seller	Admin	Owner
Audit				R	R

4.1.4 Surveys (Questionnaires and Interviews)

4.1.4.1 What type of changes could be recorded and updated in the Audit Log

What type of changes should be recorded and updated in the Audit Log?

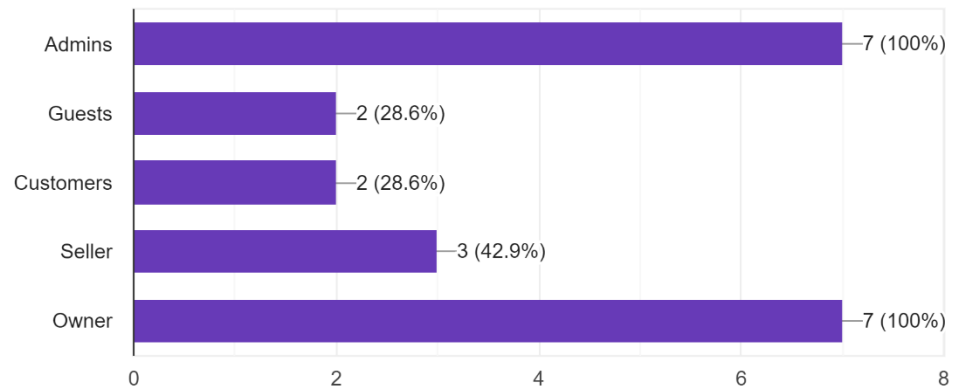
7 responses



4.1.4.2 Who should be allowed to view the log?

Who should be allowed to view the log?

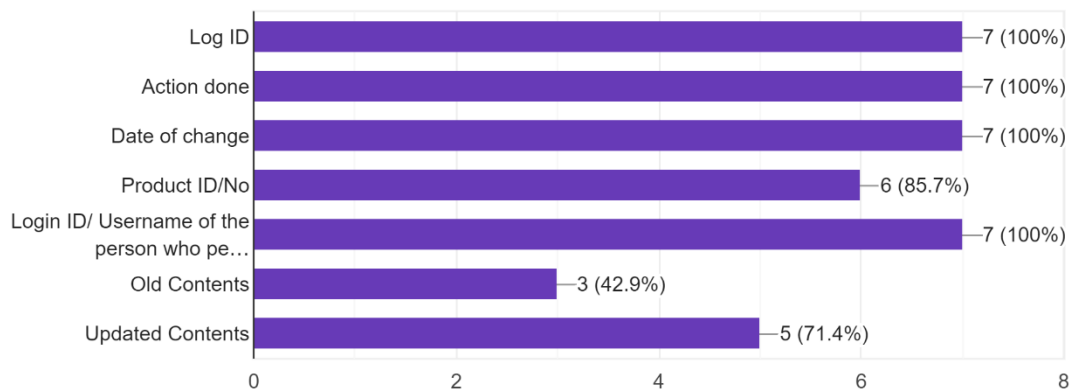
7 responses



4.1.4.3 What details should the audit log show?

What kind of details should the audit log show?

7 responses



4.1.5 Related Security Requirements

4.1.5.1 CIA

Confidentiality

For audit, only admins and the owner can see the audit logs. Other roles cannot.

Integrity

For audit, only admins, sellers and the owner can update/remove/create product listings and update it to the database. Since audit log cannot be edited or delete by anyone normally, the admins and owners who are the only ones who can see the audit logs, they are able to track everything from log in to managing of the products.

Authentication

In order to see the audit logs, the owner and admin must be logged into their accounts to gain access.

Authorization

Audit logs can only be seen by admin and owner who have access to the logs.

Auditing

An audit log is created when:

- A toy listing is created, edited or deleted
- A role is created, edited or delete
- A role is assigned to a user
- A user creates an account
- A user of any role logs in
- A user of any role fails to log in
- A user of any role buys a product
- Any user except for seller and guests leaves a review
- A customer, admin, owner deletes a review
- A review is edited
- Any user's accounts are updated/changed

4.1.5.2 Session Management

Any changes during the sessions or any changes made to the database are still recorded and updated into the audit log to trace.

4.1.5.3 Error Management

If any user without authorization tries to access the audit log, they will be directed to the access denied page and be unable to view the audit log. As for guest, since they do not have an account registered or are not logged in, they will be directed to the log in page instead.

4.2 Secure Software Design

4.2.1 Related Security Design Principles

4.2.1.1 Least Privilege

Only admins can view audit logs but are unable to edit, change or remove the logs. However, the owner is still allowed to view the logs as the owner is allowed access to everything.

4.2.1.2 Separation of Duties

A seller is not allowed to buy products and edit or delete the customers' reviews to avoid biased ratings. It also allows room for integrity of the seller's business.

4.2.1.3 Fail Safe

If owner account is lost, only admins can access audit logs and manage roles.

4.2.2 Attack Surface Evaluation

Audit Log Manipulation

The hacker can hack the website in order to gain access to the audit logs. After which, they will insert, modify or delete files from the audit log. By doing so, they can cover up any attack they have carried out or any changes made to the website or software. The hacker will only be able to do this due to inefficient or insufficient access controls of the log files or exploiting the log in section. They may also steal cookies and log into the account with admin or owner role. In order to mitigate audit log manipulation, the audit logs must strictly only be for admins and the owner to view. Any other roles should be denied access to the audit page. If hackers can hack into the software or website, the integrity of the platform is affected. Hackers can change anything in the platform and remove or change the entries in the log in order to hide the changes or attacks they have made. And these changes made could be modifying the database or adding malicious codes to certain pages so that the code will be able to run when someone runs the page.

4.2.3 Threat Modeling

When hackers are able to hack into the platform and gain access to the audit log they can practically do everything without the admins, owner finding out about which account was hacked or how they were attacked as the hacker can manipulate and cover up their tracks.

Data access control matrix from 4.1.2.2:

Data	Roles				
	Guest	Customer	Seller	Admin	Owner
Audit				R	R

Entry points:

- LogOn Page
- Search bar
- Review Page
- Shopping Cart
- Adding Billing information page

Exit Points:

- Toy Index Page
- Search Results
- Review Page
- Payment Processing
- Credit card verification

Stride table:

STRIDE List	Identified Threats
Spoofing	<ul style="list-style-type: none"> • Cookie Replay • Session Hijacking
Tampering	<ul style="list-style-type: none"> • Audit Log Manipulation • Cross Site Scripting • SQL Injection
Repudiation	<ul style="list-style-type: none"> • Audit Log deletion/ modification
Information Disclosure	<ul style="list-style-type: none"> • Output Caching
Denial of Service	<ul style="list-style-type: none"> • Website defacement
Elevation of Privilege	<ul style="list-style-type: none"> • Logic Flaw • Accounts can access previously inaccessible pages

DREAD table:

Threat	D	R	E	A	D	AverageRank
Cookie Replay	3	2	2	3	2	2.4
Session Hijacking	3	2	2	3	3	2.6
Audit Log Manipulation	2	2	1	1	1	1.4
Cross Site Scripting	3	2	2	3	2	2.4
SQL Injection	3	2	2	3	2	2.4
Audit Log Deletion/Modification	0	0	1	1	1	0.6
Output Caching	3	3	2	3	3	2.8
Website Defacement	2	2	1	3	3	2.2
Logic Flaw	2	1	1	2	1	1.4
Accounts can access previously inaccessible pages	2	1	1	2	2	1.6

4.3 Secure Software Coding

4.3.1 Related Defensive Coding Practices

4.3.1.1 Audit Trail

Audit Trail has been implemented under audit feature.

An Audit trail is a record that provides evidence of the sequence of activities that have been carried out or done at any time with information on a specific action or event. Most audit trails also include who performed the action. In cases where an attack has occurred, the audit trail is used to find out what attack and where it has been implemented, provide evidence and used to detect and prevent attacks.

Registration of users will also have an audit trail logged in into the Audit Page.

4.3.1.2 Exception Management

In order to defend information exception management is implemented to hide error messages from hackers. Error messages are one of the first sources an attacker will look to determine the information about the software. If the error is not hidden, it could potentially reveal important information as the stack traces, stored procedure name, file location and for log in pages, the login name and more. Thus, this has been implemented to conceal this information in order to prevent it from being revealed.

-insert images of access denied-

4.4 Secure Software Testing

4.4.1 Test Strategy

4.4.1.2 Features to be tested

4.4.1.3 Audit Log Page

Displaying a list of audits which are recorded and stored in the database and also allowing owner to keep track of the actions taken by all the users.

4.4.1.4

Test Methods

Functional Testing

- Logic Testing

Security Test

- White Box Testing

Software Security Test

- Testing for Non-Repudiation Controls

4.4.2 Test Plan

- Logic Testing:
 - Only owner can strictly view the Audit log. Unable to create or delete any audit logs.
- White Box Testing
 - Error Handling
 - The guests and customers are not able to access the audit log page and a page should be displayed which says "Access Denied".
- Testing for non-repudiation controls
 - The audit logs accurately show the users and their actions
 - Verifying that the security and integrity of audit logs
 - Verify the confidentiality of audit logs
 - Verify that the audit logs stay even after a certain period.

4.4.3 Test cases

Case 1:

View Audit log

- Attempt to mess around with site.

Index				
Create New				
Audit Action	Role of User	Performed By	Date/Time Stamp	Listing ID
Failed Login		admin1@gmail.com	08/06/2020 06:30 pm	Details Delete Edit
Assigned owner@gmail.com Owner Role	Owner	owner@gmail.com	08/06/2020 07:10 pm	Details Delete Edit
Add Toy Listing	Owner	owner@gmail.com	08/06/2020 11:11 am	1 Details Delete Edit
Created Admin Role	Owner	owner@gmail.com	08/06/2020 11:12 am	Details Delete Edit
Assigned admin1@gmail.com Admin Role	Owner	owner@gmail.com	08/06/2020 07:12 pm	Details Delete Edit
Add Toy Listing	Admin	admin1@gmail.com	08/07/2020 09:47 am	2 Details Delete Edit
Add Toy Listing	Admin	admin1@gmail.com	08/07/2020 09:56 am	3 Details Delete Edit

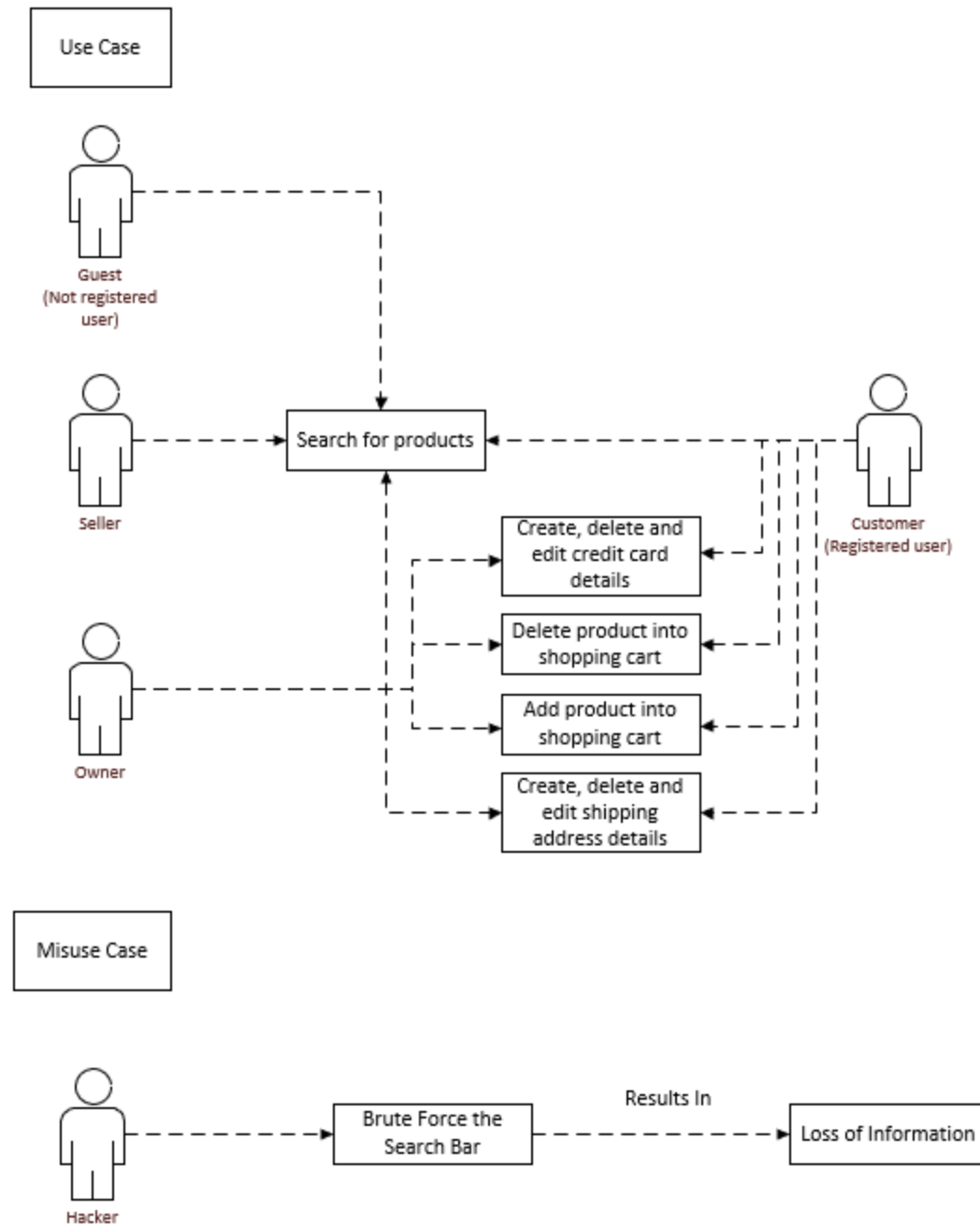
4.5 Security Conclusion of the Feature

The audit log was coded and tested to the best of our abilities to fulfil the CIA security traits. As the audit log does not take in much inputs, not much can be tested for it. We have tried and tested it vigorously to ensure it have no loopholes that could be exploited.

Feature 5: Miscellaneous Functions (Search, Shopping Cart, Payment, Error Page)

5.1 Secure Software Requirements

5.1.1 Use Case and Misuse Case Modeling (done)



5.1.2 Data Classification & Labelling

Data	Impact Upon Loss		
	Confidentiality	Integrity	Availability
PaymentDetails			
cusid	High - Restricted	High	High - Critical
ShippingAddress	High - Restricted	High	High - Essential
PostalCode	High - Restricted	High	High - Essential
Token	High - Restricted	High	High - Essential
Total	Low - Confidential	Low	Low - Essential
Email	High - Restricted	High	High - Essential

- Shopping cart does not have its own database object as the information contained within the shopping cart is similar to the product. (is this a good enough reason?)

5.1.3 Subject/Object Matrix

Data	User Roles			
	Guests	Customer	Seller	Owner
Search	CR	CR	R	CR
Shopping Cart	-	CRUD	-	CRUD
Payment Info	-	CRUD	-	CRUD

5.1.4 Data Ownership

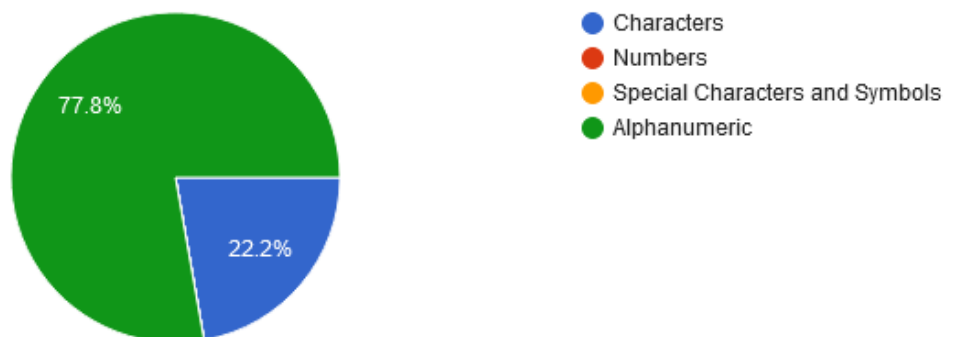
- Owner:
Owner has all rights and access to the database, which contains the credit card details and information on all users.
- Seller:
It has strictly does not have access to the shopping cart, payment information of customers and database. It has access to the search function located on the Toys Page.
- Customer:
Customers can add payment information, change personal information, use the search function on Toys Page, has access to shopping cart and payment page. But strictly unable to create, delete, modify any products.
- Guest:
Guests are not able to access the payment page as they must be a registered customer first.

5.1.5 Surveys (Questionnaires and Interviews)

The people who are involved in this questionnaire are students from Ngee Ann Polytechnic.

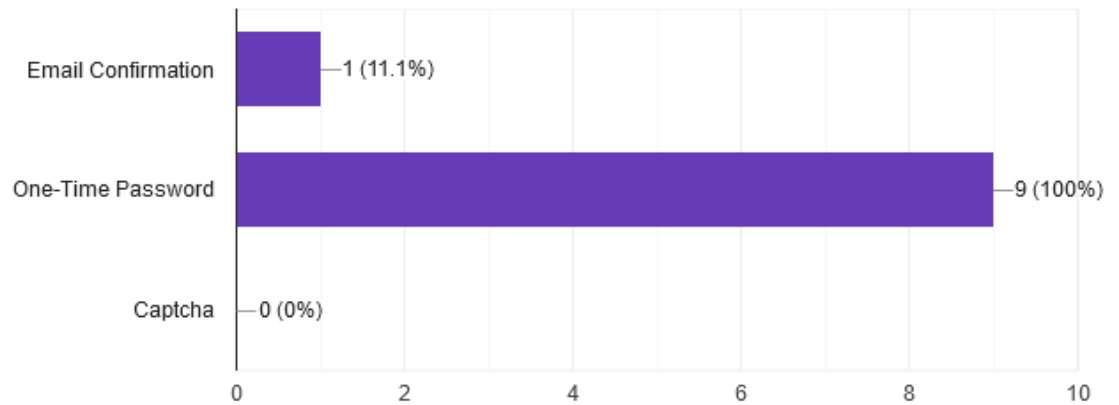
- What types of inputs should be allowed in the search field?

9 responses



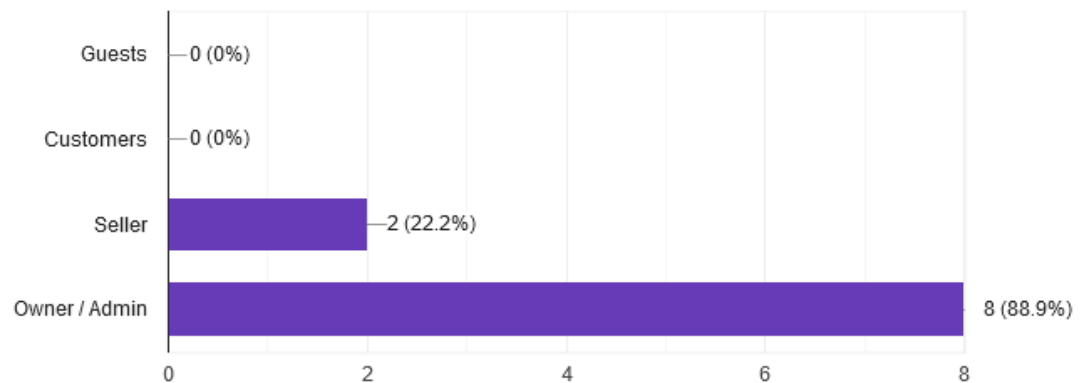
- What type of verification should be done when making a purchase?

9 responses



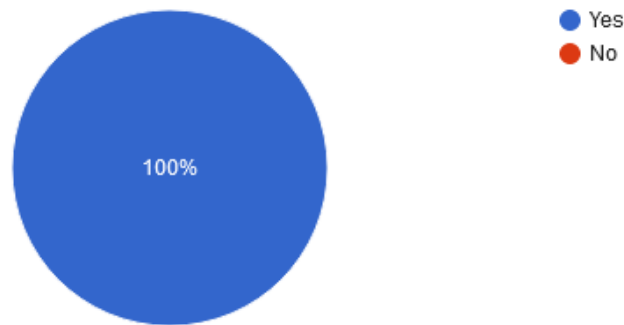
- Who has access to the credit cards detail after it is keyed into the website for payment?

9 responses



- Are customers allowed to remove their items from the shopping cart?

9 responses



5.1.6 Relevant Security Concepts and Management

- **CIA**
 - **Confidentiality**
 - Payment Information such as Credit Card information and Address of customers are not accessible to other customers.
 - **Integrity**
 - Credit Card information is only accessible by Owner. Owner can change the information as it is in the database.
 - Input validation is in place to check for credit card validity.
 - **Availability**
 - Customers should always be able to make payment at any time of the day, on any day of the week.
 - Payment function should always be available for users to use and make payments.

- **AAA**

- **Authentication**

- Customers must register as a customer or log into an existing account before they are able to add an item into shopping cart and make payment for an item.
 - Seller must log into a seller account before they can create product listings, view, modify and delete listings.
 - Owner must log into the administrator account and have full access.
 - All users must verify their identity by using either email confirmation, OTP verification to their mobile devices or by enabling 2-factor authentication (2FA), upon initial entry to their account.

- **Authorization**

- Customers are strictly only allowed to use the shopping cart, payment page and search functions.
 - Sellers are strictly not allowed to access shopping cart and payment page.
 - Only an Owner can access the database and verify purchase of product of the customer.

- **Session Management**

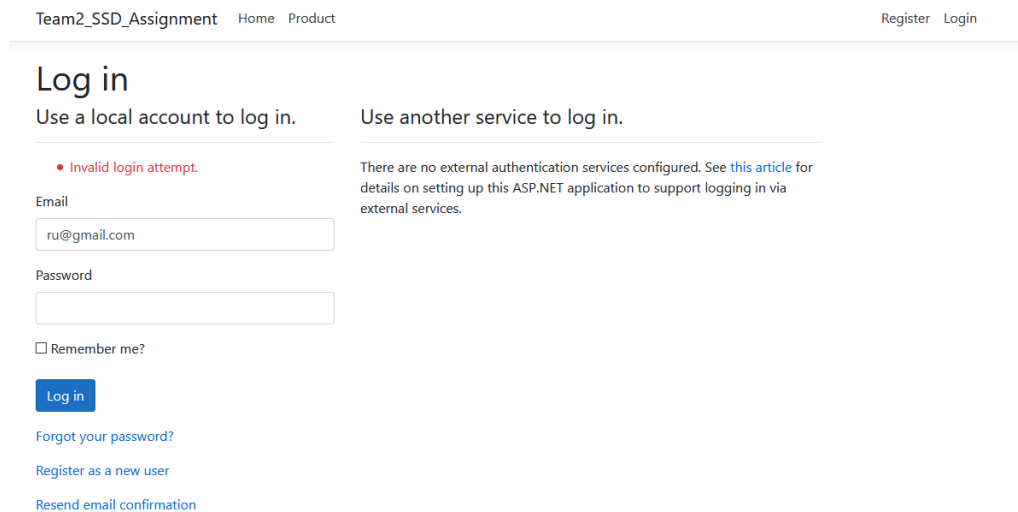
- OTP will reset after 5 mins of the payment page being opened.
 - After the 5 mins, the user will not be able to make payment as the OTP would have expired.
 - We were not able to implement this feature due to lack of time.

- **Error Management**

- When a user just registered for an account, the user must first verify their email address before they are allowed entry into the webpage. Otherwise, an error will occur, and the image shown below will be displayed.



-
- The guest has to confirm his email before logging in. If the guest attempts to log in without verifying the email address, this error message will show.



-

5.2 Secure Software Design

5.2.1 Secure Software Design Principles

- Least Privilege
 - The lowest tier of authorization was given to the different users to accomplish each of their necessary tasks. An example would be that customers are not given the rights to view the personal information of other users due to privacy policies and they would have no use knowing names and addresses of other customers.
- Defense-in-Depth
 - Several methods to ensure secure coding implementations were made such that there would be layers of security within the website application to prevent any single point of failure.
- Fail Safe
 - When an error occurs, the appropriate pages showing error or exception messages will be displayed instead of the default error and exception screen to prevent the vulnerability within the software. An error message will be shown if there is an error while an exception page will be shown when a user is not authorized to access the page.
 - No users will be taken to action when an error page or an exception page is shown as to maintain the integrity of the traits “CIA” for the website application.
- Economy of Mechanism
 - The website application is built in a manner that is simple and clear such that users will be able to easily locate each function.
 - This is also built as simple as possible but with necessary security to reduce attack surface on any pages within the application.
 - While ensuring security, user accessibility and usability is ensured and tested such that it does not hinder from a great experience.

- Complete Mediation
 - Authorization is checked when a user sends a request such as making payment and adding an item into the shopping cart.
 - Only when user is verified and authorized will then the request be processed and granted by the server.

- Psychological Acceptability
 - The pages used should be readable and simple to understand, using no special characters and not requiring special characters as inputs.
 - The implementation of shopping cart items does not require manual input and would be processed by a press of a mouse button. This increases usability and ease for the customers to browse the website application.

5.2.2 Attack Surface Evaluation

Surface	Is it exploitable by the hacker?	Mitigation to an attack by hacker.
Search Function	Yes. The input boxes allow hackers to input malicious code into the function.	Input validation and restriction on certain characters in the input boxes, prevents SQL Injection and Cross-site Scripting attacks.
Payment Page	Yes. The input boxes for payment allows hackers to input malicious code into the function.	Input validation and restriction on certain characters in the input field prevents SQL Injection and Cross-site scripting attacks.
Payment Info Page	No. This page does not allow any input.	-
Shopping Cart Page	No. This page only allows users to view the item and	-

	does not have any input fields.	
My Account Page	Yes. This page allows some input by the user such as email address.	Input validation and restrictions have been set to prevent certain characters in the input field to prevent SQL Injection and Cross-site scripting attack.
OTP Page (not yet)	Yes. The OTP textbox allows input of malicious code.	Have strong input validation and validate all user inputs and filter for characters and SQL content
Product Page	Yes. The creation of products allows for hackers to input malicious codes into the page.	Input validation and restriction on certain characters in the input field prevents SQL Injection and Cross-site scripting.
Credit Card Page	Yes. The addition of credit card allows input of malicious code.	Have strong input validation and validate all user inputs and filter for characters and SQL content
Address Page	Yes. The addition of shipping address allows input of malicious code.	Have strong input validation and validate all user inputs and filter for characters and SQL content
Product Quantity Page	Yes. The text boxes allow input of malicious code.	Have strong input validation and validate all user inputs and filter for characters and SQL content

5.3 Threat Modeling

5.3.1 Subjects & Objects of the system

- Guests are not able to add an item into a shopping cart, make payments or add personal credentials as they do not have an account.
- Customers are able to add an item into a shopping cart, make payment for their items, add personal credentials and more as they have a registered account.
- Seller can strictly view feedback, can use search function, but is not able to create a shopping cart or make payments for any products.
- Owner can manage add items into shopping cart, make payment for the product, add personal credentials and strictly view the audit log only.

5.3.2 Data Access Control Matrix

	Guest	Customer	Seller	Owner
Shopping Cart	-	C R U D	-	C R U D
Payment Page	-	C R U D	-	C R U D
Search Function	C D	C D	C D	C D
Audit Log	-	-	-	R

- **Technologies used to build the application**
 - Visual Studio
 - SQL Server
 - ASP.NET, Razor Pages

- **External dependencies**

- Stripe

5.3.3 Trust Boundaries

- It exists between external (Internet) and Demilitarized Zone (DMZ) Network.
- It exists between Demilitarized Zone (DMZ) Network and Internal (Intranet) Zones.

5.3.4 Entry Points

Entry Point	Description of Entry Point	Accessibility limits
"My Account" Page	Page where users can modify their personal credentials and information (shipping address, password, credit card number, etc.)	Customers, Sellers, Owner
Role Page	This is a page only the owner can access and handle. It is also the page where roles are created.	Owner
Role Management Page	This page is where owner can use to assign, view, and delete user roles.	Owner
Payment Page	Users enter personal credentials and make payment here.	Customers
Search Function	A function where users can	Guests, Customers,

	enter an input in string format to filter the search results.	Sellers, Owner
--	---	----------------

5.3.5 Exit Points

Exit Point	Description of Exit Points	Role Accessibility
Search Results Page	This page returns the results of the keywords searched by the user to filter the toys.	Guests, Customers, Sellers, Owner

5.3.6 General data flow

- Customers
 - An existing customer logs in □ Customer views products □ Product is added to cart □ Customer is redirected to shopping cart page □ Customer press Payment à Customer is redirected to Payment page à Customer enters credentials à Customer order is being submitted □ Customer is redirected to payment page
- Seller
 - Seller registers for an account à Owner assigns seller to a seller authorization

- Owner
 - Owner logs in □ Owner goes to Manage Roles Page
 - Owner logs in □ Owner views products □ Product is added to cart □ Owner is redirected to shopping cart page □ Owner press Payment □ Owner is redirected to Payment page □ Owner enters credentials □ Owner order is being submitted □ Owner is redirected to payment page

5.3.7 Privileged functionality

- The codes that allow the increase in privileges or execution of operations are identified.
- Guests have functions such as product management (read) are identified.
- Customers has product management (read), account management (create, update, delete) and feedback management (create, update, delete) functions identified.
- The seller functions consist of product management (read, create, edit, delete) are identified.
- The owner has functions which consists of product management (read), account management (create, update and delete), feedback management (create, update, delete) as well as audit management (read) are identified.

5.3.8 Mis-Actors

- Internal and external threats and identified and introduced.
 - Examples of Human Mis-actors: External hacker, Hacktivist group, Rouge administrator
 - Examples of Non-human Mis-actors: Internal running process making unauthorized changes, malware.

5.3.9 Potential and applicable threats through STRIDE

STRIDE	Security Property	Identified threat
Spoofing	Authentication	<ul style="list-style-type: none"> • Brute Force Authentication • Cookie Replay • Session Hijacking • CSRF
Tampering	Integrity	<ul style="list-style-type: none"> • Cross-Site Scripting (XSS) • SQL Injection
Repudiation	Non-repudiation	<ul style="list-style-type: none"> • Audit Log Manipulation • Insecure Backup
Information Disclosure	Confidentiality	<ul style="list-style-type: none"> • Eavesdropping Verbose Exception • Output Caching • Path Traversal
Denial of Service	Availability	<ul style="list-style-type: none"> • Website Defacement
Elevation of Privilege	Authorization	<ul style="list-style-type: none"> • Logic Flaw

5.4 Identifying, Prioritizing and Implementing Controls

5.4.1 Delphi Ranking

Threat	D	R	E	A	DI	Total	Result
SQL Injection	3	3	2	3	2	13	High
Cross Site Scripting	3	3	3	3	3	15	High
Cookie Replay	3	2	2	2	2	11	Medium
Session Hijacking	2	2	2	1	3	10	Medium
Cross Site Request Forgery	3	1	1	1	1	7	Low
Verbose Exception	2	1	2	3	1	9	Medium
Brute Force	2	1	1	3	2	9	Medium
Eavesdropping	2	1	1	2	2	8	Medium
Insecure Backup	2	1	1	2	2	8	Medium
Audit Log Deletion	1	1	1	1	3	7	Low
Output Caching	3	3	2	3	3	14	High
Website Defacement	3	2	1	3	3	12	High
Logic Flaws	3	1	1	2	1	8	Medium

5.4.2 Probability x Impact (P x I) Ranking

Threat	Probability of Occurrence (P)			Business Impact		P	I	Risk
	R	E	DI	D	A	(R + E + DI)	(D + A)	P * I
SQL Injection	3	2	2	3	3	7	6	42
Cross-Site Scripting	3	3	3	3	3	9	6	54
Cookie Replay	2	2	2	3	2	6	5	30
Session Hijacking	2	2	3	2	1	7	3	21
CSRF	1	1	1	3	1	3	4	12
Verbose Exception	1	2	1	2	3	4	5	20
Brute Force	1	1	2	2	3	4	5	20
Eavesdropping	1	2	2	2	2	5	4	20
Insecure Backup	1	1	2	1	2	4	3	12
Audit Log Deletion	1	1	3	1	1	5	2	10
Output Caching	3	2	3	3	3	8	6	48
Website Defacement	2	1	3	3	3	6	6	36
Logic Flaws	1	1	1	3	2	3	5	15

5.5 Documentation and Validation

After documentation of the lists of threats and controls and the residual risks involved, our group conclusively validated that the platform threat model would have the following key characteristics:

- The 3-tier application architecture model is accurate and is up to date.
- That threats are identified when crossing each trust boundary and each data element.
- Ensure that the design built is secure and is able to withstand attacks.

5.5.1 Threat Mitigation

This table shows the PI Threat level of each vulnerability and some ways to mitigate the attacks,

Threat - (P x I Rank)	Steps to mitigate threat
SQL Injection - (42)	<ul style="list-style-type: none"> • Input validation. • Parameterized Queries. • Stored Procedures. • Escaping.
Cross Site Scripting - (54)	<ul style="list-style-type: none"> • Escaping. • Input validation. • Sanitization.
Cookie Replay - (30)	<ul style="list-style-type: none"> • Cookie-less authentication. • Encrypt cookies to avoid tampering. • One-time token.
Session Hijacking - (21)	<ul style="list-style-type: none"> • Generate and use random session identifiers. • Abandons session after use. • Log off automatically after leaving the site.

	<ul style="list-style-type: none"> • Using “httpOnly” in the code to disable JavaScript code from reading the session identifier. • Setting the secure flag.
CSRF - (12)	<ul style="list-style-type: none"> • Use unique session tokens. • Use referrer origin checks. • Complete mediation. • Implement anti-CSRF token. • Use SameSite flag in cookies.
Verbose Exception - (20)	<ul style="list-style-type: none"> • Use non-verbose error message. • Fail secure.
Brute Force - (20)	<ul style="list-style-type: none"> • Don't allow weak passwords using regular expression. • Balance psychological acceptability with strong passwords.
Eavesdropping - (20)	<ul style="list-style-type: none"> • Enable data encryption. • Sniffers detection. • Disallow rogue systems.
Insecure Backup - (12)	<ul style="list-style-type: none"> • Enable data encryption. • Using SSL (Secure Socket Layer) for transport or IPsec (Internet Protocol Security) for network in-transit protection.
Audit Log Deletion - (10)	<ul style="list-style-type: none"> • Don't allow other users direct access to the database
Output Caching - (48)	<ul style="list-style-type: none"> • Don't cache credentials • Implement complete mediation
Website Defacement - (36)	<ul style="list-style-type: none"> • Load balancing and DR

	<ul style="list-style-type: none">• Disallow URL redirection
Logic Flaws - (15)	<ul style="list-style-type: none">• Design Reviews

- Each threat has been explicitly considered and controls for mitigation, acceptance, or avoidance have been identified and mapped to the threats they address.
- The residual risk of that threat is determined and formally accepted by the business owner, if the decision to accept risk is made.
- It is also important to revalidate the threat model, should the scope and attributes of the application change.

5.6 Secure Software Processes

5.6.1 Versioning

- This ensure that all members on the team is working on the most up-to-date version of the code, granting rollback if necessary. This has a secondary function, which is the audit log, displaying the changes in code and the users who pushed the changes. We coded on our individual devices and used Bitbucket as a platform to compile and update our codes for others.

5.6.2 Code Analysis

- Static Code Analysis
 - This is the process of inspecting the code in Visual Studio 2019 to figure out which codes may have the potential to cause errors. When codes are committed and pushed, the codes that are pulled by the teammates might cause errors due to incompatibility issues. The errors are indicated by a red underline in Visual Studio.

- **Dynamic Code Analysis:**
 - Dynamic inspecting involves inspecting the code during execution to figure out if there are any errors present. One example of an error is “build error”, where why the project can compile, but has issues when it tries to run. This is critical to know the errors that are still present in the project, ensuring that it can run as expected us users. The reason for these errors is perhaps certain parameters within the database not defined.

5.6.3 Code & Peer Review

- Code is tested and reviewed by other members in the team to highlight any mistakes and to correct it before proceeding further. These are not to point out mistakes but learning points to improve the application when such a mistake occurs. Ultimately, it lowers the vulnerability and errors caused by the application when it is finished.

5.7 Secure Software Coding

5.7.1 Defensive Coding Practices

- **Role Restrictions**
 - The customer is denied accessing the page which is only authorized for the owner such as Audit Page.

Team2_SSD_Assignment Home Product

Hello 25.ezrahojincheng@gmail.com!

Logout

Access denied

You do not have access to this resource.

-
- **Error and Exception Management**
 - Users will be directed to an error page, where there will be a link to return to the main page if there are any errors occurred in the page.
 - Input validation errors will not redirect users to an error page, but it will disable users from submitting the page and moving on from that page to other pages.

Team2_SSD_Assignment Home Product Hello 25.ezrahojincheng@gmail.com! Logout

Manage your account

Change your account settings

[Profile](#)
[Email](#)
[Password](#)
[External logins](#)
[Two-factor authentication](#)
[Personal data](#)

Change password

Current password

••••••••••

New password

•••

The New password must be at least 6 and at max 100 characters long.

Confirm new password

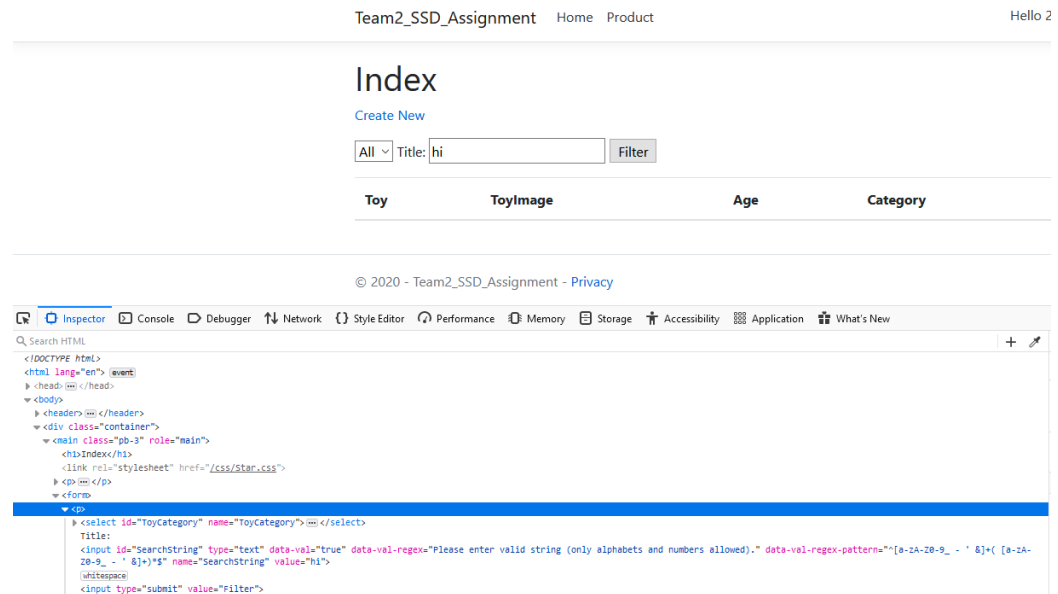
••••

The new password and confirmation password do not match.

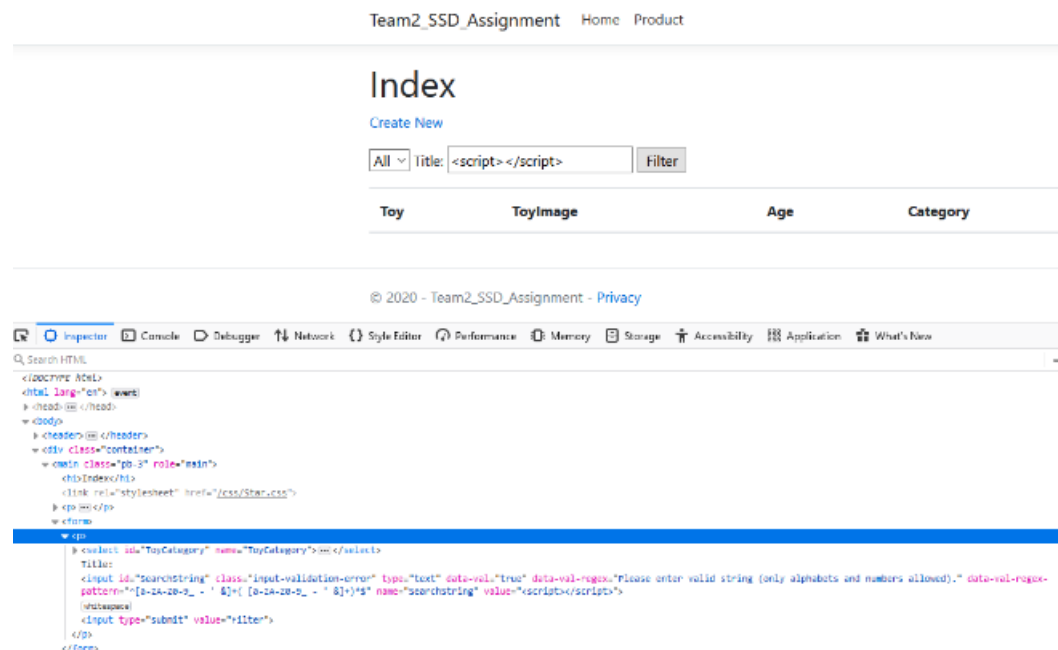
Update password

-
- Session Management
 - OTP is only set to be valid for 5 minutes. After 5 mins, the OTP will no longer be valid and as such, keying in the OTP after 5mins will not allow the payment to past through.
 - There is not enough time for us to implement this function.
- Data Input Validation
 - Any input by the user has been validated first using SQL Injection or Cross-site scripting before it is sent to the server.
 - Symbols such as !, < , > , @, =, -, _ , (,) , * , & , ^ , % , \$, ? , / , : , ; , “ , ‘ , { , } , [,] , \ , | , + , ` , ~ , # are all not allowed unless the input accepts it.

- Search bar



-
- The above picture shows the text where it shows the input was being searched.



-
- On the other hand, this picture shows that the input string was sanitized. However, there was no output to show the users of the website.

5.8 Secure Software Testing

5.8.1 Test Strategy

- Features to be Tested
 - Product Page
 - Stores the product data in the database.
 - Display list of products allow everyone to view.
 - Shopping Cart Page
 - Adding and deleting an item from shopping cart.
 - Users are only allowed to view their own shopping cart.
 - Payment Page
 - One Time Password (OTP) will be sent to the mobile phone to verify the payment.
 - Customer should be able to see their own purchased item.
 - Payment Page
 - Customers and owner can create, modify, and delete their credit card details.
 - Customers are only allowed to view their own credit card details and address.
 - Owner is only able to see the details and address of everyone from the database.
 - Owner is only able to see the password encrypted password, which is stored in the database due to the nature of this assignment. Usually, websites are not able to keep the data of the credit card even if it is hashed, as it is not safe and is dangerous if the database happens to be hacked.
 - Search bar
 - Symbol such as !, < , > , @ , = , - , _ , (,) , * , & , ^ , % , \$, ? , / , : , ; , " , ' , { , } , [,] , \ , | , + , ` , ~ , # should not be allowed in the search function.

5.8.2 Test Plan

- Functional Test
 - Unit Testing:
 - This testing method is conducted by the developers to ensure that all the parts of the software is functioning properly. Since there are many functions such as Shopping Cart and Payment, unit testing is the testing of these individual during the coding phases of the software development. This test conducted to test the quality of code issues as well. It can uncover inefficiencies, cyclomatic complexities and vulnerabilities in the code that might cause an issue further down the line.
 - Integration Testing:
 - This test is conducted to ensure the software is functionally reliable, resilient, and recoverable. With the Shopping Cart and Payment page passing the Unit Testing, this testing method will test whether they are able to work together. This test is conducted after unit testing to validate the functionality of the software. The security portion of the sum of all parts is also tested as although unit testing security may pass, but if integration testing security fails, then upon integration of this program, there will be issues that pop up. Hence, the reason why integration testing is crucial.

- Security Test
 - Black Box Testing:
 - This test is conducted when the tester does not have any knowledge of the internal workings of the software. All the documents such as design, configuration information and use and misuse cases of the source files are not available to the tester when conducting black box testing. This is when the software is really put to the test of resiliency as it responds differently to how the tester tests the software. Black Box testing is also known as the behavioral analysis test.
 - White Box Testing:
 - This test is also known as glass box testing as the tester has the access to all the design codes, use and misuse cases more. The tester has complete knowledge about the inner workings of the software, and hence can target the portions to attack specifically. This test heavily tests the structural analysis of the software's security.
 - Fuzzing:
 - Also known as Fault Injection Testing, this testing method brute force type of software injects faults into the software and observes the behavior of the software afterwards. This test roughly indicates the effectiveness and the extent of mitigation input validation placed by the developers can stop attackers from injecting malicious codes and faults into the software after release.
 - Finding code defects and security bugs are one of the many functions which can lead it to discover remote code execution, unhandled exceptions and hanging threads that can cause a DoS attack to be executed. Fuzzing can also be used as a black box or white box methodology.

- Software Security Test:
 - Testing for Input Validation:
 - All the input fields can be protected by input validation, which protects the software from SQL Injection and Cross Site Scripting attacks due to vulnerabilities.
 - All fields can be tested such as search function input field.
 - Testing for Scripting Attacks Controls:
 - Scripting attacks can be done by hackers, due to the lack of sanitization of codes and input fields from the developer. This can be mitigated if the input fields are properly sanitized and testing of input fields are done beforehand.
 - Testing for Error and Exception Handling Controls (Failure Testing):
 - The software is prone to failure which is caused by an accidental user error or an intentional attack. Thus, the software must be thoroughly checked and tested for the quality and reliability, so it does not crash and fail when users are using it.
 - Testing for Privileges Escalations Controls
 - This ensures that users are not allowed higher access and obtain more resources and functionality than previously allocated and authorized by the owner.

5.8.3 Test Cases

Test Case ID	Test Scenario	Test Case	Test Steps	Expected Result	Actual Result
A1	Seller add Product into shopping cart	Attempt to add product into shopping cart	1. Seller logs in 2. Seller adds product into shopping cart	Not possible. As there is no button for the seller to add the item into the cart.	Not possible. As there is no "Add to cart" button for the seller to add item into cart
A2	Seller attempt to add payment details	Attempt to add payment details	1. Seller logs in 2. Seller adds payment details into shopping cart	Not possible. Since there is no button for the seller to add the payment detail.	Not possible. As there was no way a seller could add payment details unless seller had access to shopping cart.

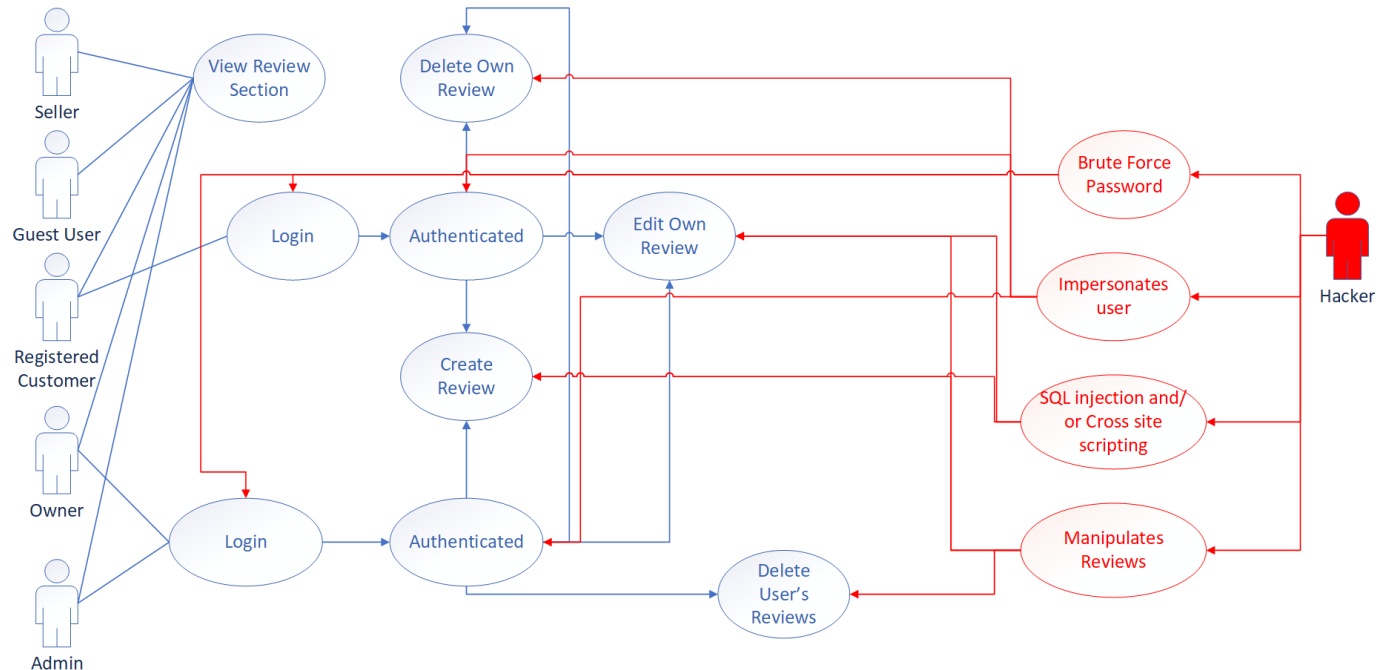
5.9 Security Conclusion of the Feature

After testing the web pages, ensuring that each of the features fulfill the Secure Software Requirements; the search function, the product pages where it displays the toys, the shopping cart and payment page are all secure. As for the products, the credit cards and other security software implementations are data input validation, exception, error management and role restriction. These features prevent malicious attacks such as SQL Injection, which prevents users from gaining more access to the back-side of the website and also denies unauthorized users. IN regard to the payment page, the security software implemented is session management, where a one-time password will be invalid after 5 minutes. The security software implemented in these functions will increase the security and usability of the website.

Feature 6: Review

6.1 Secure Software Requirements

6.1.1 Use Case and Misuse Case Modeling





6.1.2.1 Data Labelling

The page will first display the Rating_ID which is hidden, followed by the review of the user, the uploaded image, the user name of the customer who created the review (if no username, the default is email), the toy name, the star rating and the date of review. In the database the product ID (ListID) is taken in as well to reference to the database to extract the name of the product. The star rating can only have a max of 5 stars and minimum of 1 star. The uploaded file cannot be more than 2 mb as well and cannot be anything other than png or jpeg.

Display Example:

Rating ID (Hidden)	[Blank] (Toy image)	Review (shows first 20 character)	Reviewed By	Toy Name	Star Rating	Date of review

2		Good	peter	PlayDoh	5	08/06/2020 07:35 pm
1		Bad product	owen12345	Uno	1	08/06/2020 07:22 pm

6.1.2.2 Data Classification

Data	Impact upon loss		
Review Page data	Confidentiality	Integrity	Availability
Rating_ID	Low	High	Low
Review	Low	High	Low
Username	Low	High	Low
RatingStar	Low	High	Low
DateReview	Low	High	Low
ListID(hidden)	Low	High	Low
namaetoy	Low	High	Low
ImageToy	Low	High	Low

6.1.3 Subject/Object Matrix

Data	Roles				
	Guest	Customer	Seller	Admin	Owner
Review	R	R,W	R	R,W	R,W

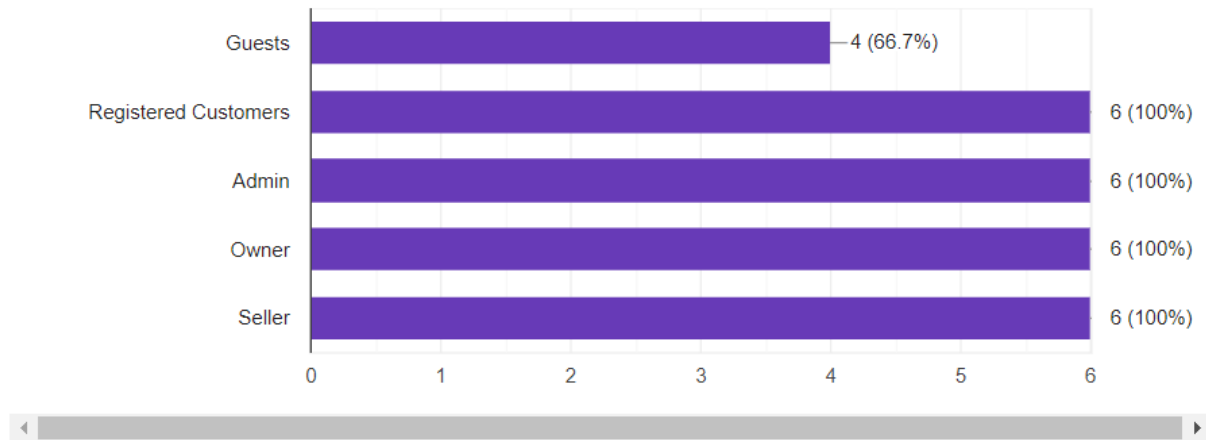
6.1.4 Surveys (Questionnaires and Interviews)

6.1.4.1 Who are allowed to see the review page?

Who are allowed to see the review page?



6 responses

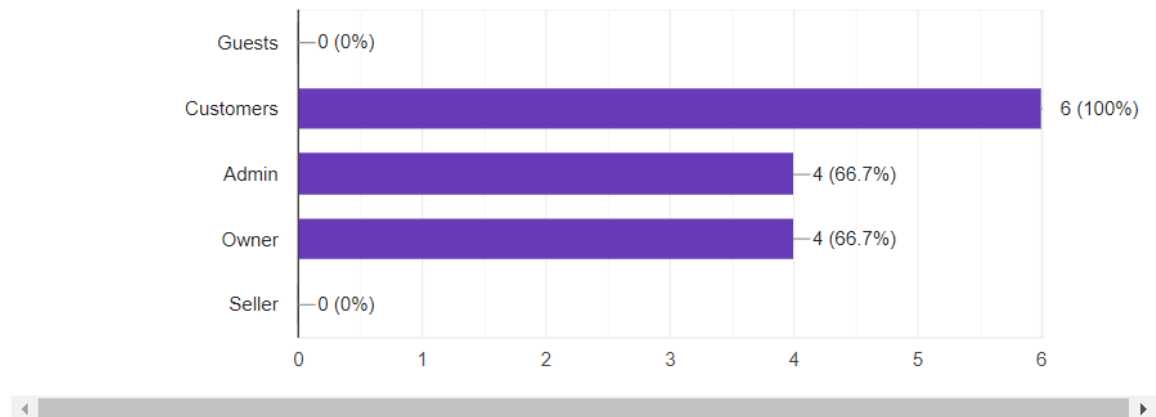


6.1.4.2 Who can create reviews?

Who can create reviews ?



6 responses

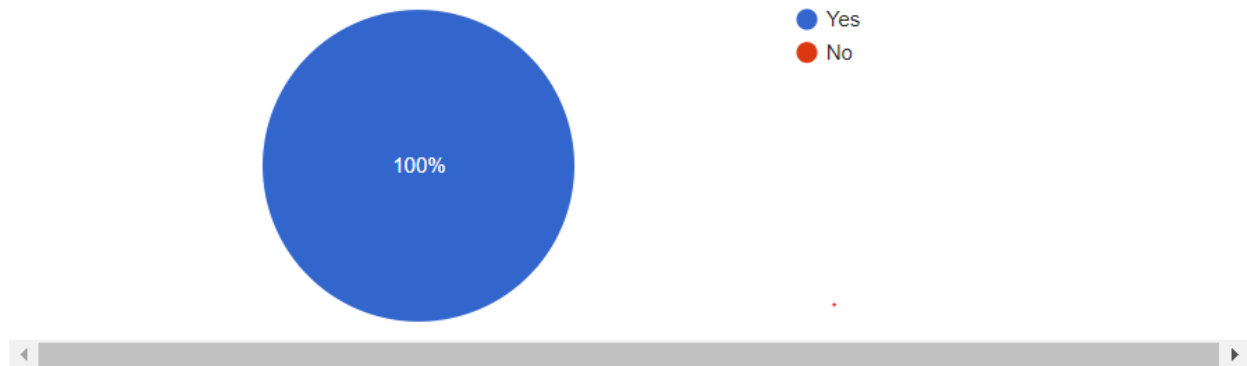


6.1.4.3 Are they allowed to edit and delete their own reviews?

Are they allowed to edit and delete their own reviews?



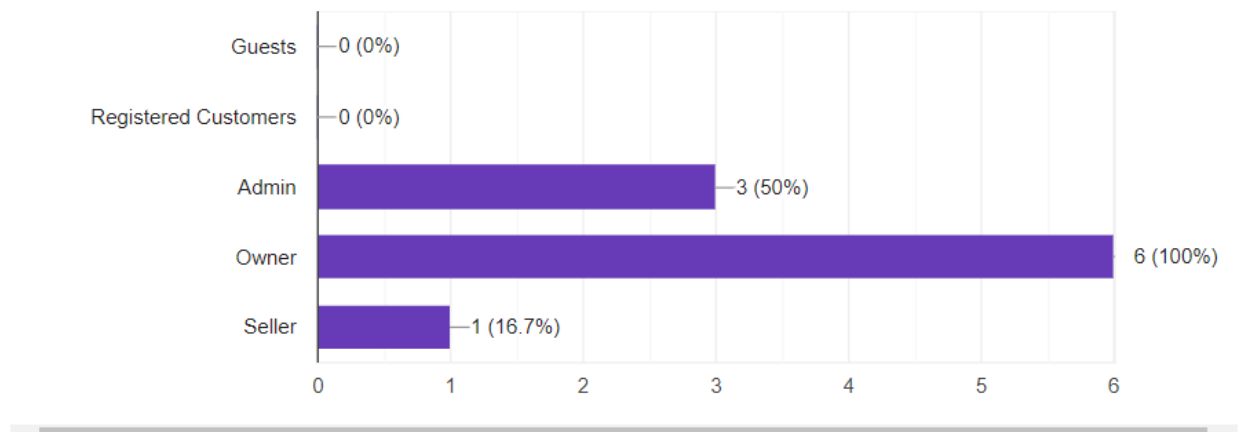
6 responses



6.1.4.4 Who can delete user reviews?

Who can delete user reviews?

6 responses



6.1.5 Related Security Requirements

6.1.5.1 CIA

Confidentiality

User reviews are not confidential as reviews are meant for all to see. However, information such as UserID or product ID are not to be seen and can only be seen in the database by the owner.

Integrity

For review, only the customers, admin and owner who have purchased the product can create the reviews. They are also only allowed to edit and delete only their own reviews except for the owner being able to delete other users' reviews. This is done to avoid letting the seller manipulate their product's ratings and reviews or from users who are trying to deliberately bring down the ratings of the product. Owners and admins are only allowed to create reviews once they have purchased a product. Otherwise the owner can only delete a review and admin cannot do anything else, regardless of the role.

Availability

The review page should always be available for everyone to view in order to help them with decision making when purchasing a toy or product. As for owner, it should be available for them to delete malicious comments as well, to prevent racial slurs, malicious reviews or anything else inappropriate as the review page can be seen by anyone.

Authentication

In order to created reviews, edit reviews or delete reviews, the user must first log in. Else, they can only view the reviews.

Authorization

Review Page can be seen by everyone, but creation, deletion and editing of own reviews are strictly only for owner, registered customers and admin. As for deletion of user reviews, only owner is allowed.

Auditing

Auditing is done to track user's actions incase malicious attacks were to occur from the review section and the owner/admin can track the actions.

An audit log is created when:

- A user creates a review
- A user deletes their own review
- A user edits their review
- The owner deletes a review

6.1.5.2 Session Management

Existing Users must first log in before creating, editing or deleting a review. This is done to prevent spams, malicious comments, etc. And also for authentication in order to ensure that the user is an admin , owner or customer who had purchased the product and are given permission to add or change any reviews.

6.1.5.3 Error Management

When a user tries to access a page that they do not have permission to open, they will be redirected to another page that only shows "Access denied". This is done to prevent the accidentally leak of the stack, query, etc. If the code does not run, they will be thrown into an error message page instead. If they can see the page, the hackers will be able to gain information from the error message and launch a possible attack such as cross-site scripting attacks or SQL injection.

6.2 Secure Software Design

6.2.1 Related Security Design Principles

6.2.1.1 Least Privilege

Users are given different privileges depending on their level of authorization. A guest is given the least privilege and are only allowed to view the page. Customers and admins are given the basic privileges, which is to create, edit, read and delete their own reviews once they have purchased the product. Owners are given the same permissions as well as the permission to access and modify the database and to also delete the users' reviews.

6.2.1.2 Separation of Duties

A seller is not allowed to buy products and edit or delete the customers' reviews to avoid biased ratings. It also allows room for integrity of the seller's business. An admin can only manage everything else other than the database and reviews. Owner is allowed to manage everything but not allowed to randomly create reviews, for products they have never purchased, for integrity purposes

6.2.1.3 Fail Safe

Guests, who do not have permission to create, edit or delete reviews, will be redirected to the access denied page. Customers and admins will also be redirected if they try to delete another person's review or if they try to edit another customer's review. As for owner, they only be redirected, if they do not purchase the product.

6.2.2 Attack Surface Evaluation

There are many attack surfaces that a hacker can exploit

- Cross Site Scripting (XSS):
 - Special symbols and characters can easily be used to gain unauthorized access to the database.
 - To mitigate cross site scripting, guests' and customers' input that are typed in the input field the input is sanitized and validated.
 - Another way to mitigate is to enforce strict rules on characters used and encoding.
- SQL Injection
 - This uses SQL statements to carry out a specific command on the database which is not intended by the owner of the website.
 - To mitigate an SQL Injection attack, input validation is used to filter out special characters. These characters can be symbols, characters or even punctuation.
 - Another way to mitigate is using error pages to redirect errors from exposing the issue caused by the database. This might reveal vulnerable and critical information to the hackers.

Surface	Is it exploitable by the hacker?	Mitigation to an attack by hacker.
Review Page	Yes. The input boxes allow hackers to input malicious code into the function.	Input validation and restriction on certain characters in the input boxes, prevents SQL Injection and Cross-site Scripting attacks.
Error Page	No. There are no input fields in the error page for hackers to input malicious codes.	-

6.2.3 Threat Modeling

Data access control matrix from 4.1.2.2:

Data	Roles				
	Guest	Customer	Seller	Admin	Owner
Own Reviews		C,R,U,D		C,R,U,D	C,R,U,D
Other use Reviews	R	R	R	R	R,D

Entry points:

- LogOn Page
- Search bar
- Creating review Page
- Review Edit Page
- Shopping Cart
- Adding Billing information page
- Image Uploading

Exit Points:

- Toy Index Page
- Search Results
- Review Index Page
- Review Details Page
- Payment Processing
- Credit card verification

Stride table:

STRIDE List	Identified Threats
Spoofing	<ul style="list-style-type: none"> • Cookie Replay • Session Hijacking • CSRF
Tampering	<ul style="list-style-type: none"> • Audit Log Manipulation • Cross Site Scripting • SQL Injection
Repudiation	<ul style="list-style-type: none"> • Audit Log deletion/ modification
Information Disclosure	<ul style="list-style-type: none"> • Output Caching
Denial of Service	<ul style="list-style-type: none"> • Website defacement
Elevation of Privilege	<ul style="list-style-type: none"> • Logic Flaw • Accounts can access previously inaccessible pages

DREAD table:

Threat	D	R	E	A	DI	AverageRank
Cookie Replay	3	2	2	3	2	2.4
Session Hijacking	3	2	2	3	3	2.6
Audit Log Manipulation	2	2	1	1	1	1.4
Cross Site Scripting	3	2	2	3	2	2.4
SQL Injection	3	2	2	3	2	2.4
Audit Log Deletion/Modification	0	0	1	1	1	0.6
Output Caching	3	3	2	3	3	2.8
Website Defacement	2	2	1	3	3	2.2
Logic Flaw	2	1	1	2	1	1.4
Accounts can access previously inaccessible pages	2	1	1	2	2	1.6
CSRF	3	1	1	1	1	1.4

Delphi Ranking:

	Probability of occurrence (P)			Business impact (I)		P	I	Risk
Threat	R	E	DI	D	A	R+E+DI	D+A	P X I
Cookie Replay	2	2	2	2	3	6	5	30
Session Hijacking	2	2	3	3	3	7	6	42
Audit Log Manipulation	2	1	1	1	1	4	2	8
Cross Site Scripting	2	2	2	2	3	6	5	30
SQL Injection	2	2	2	2	3	6	5	30
Audit Log Deletion/Modification	0	1	1	1	1	2	2	4
Output Caching	3	2	3	3	3	8	6	48
Website Defacement	2	1	3	3	3	6	6	36
Logic Flaw	1	1	1	1	2	3	3	9
Accounts can access previously inaccessible pages	1	1	2	2	2	4	4	16
CSRF	1	1	1	1	1	3	2	6

6.3 Secure Software Coding

6.3.1 Related Defensive Coding Practices

6.3.1.1 Exception Management

In order to defend information exception management is implemented to hide error messages from hackers. Error messages are one of the first sources an attacker will look to determine the information about the software. If the error is not hidden, it could potentially reveal important information as the stack traces, stored procedure name, file location and for log in pages, the login name and more. Thus, this has been implemented to conceal this information in order to prevent it from being revealed.

6.3.1.2 Input Validation

Any input by the users is checked and assumed as a malicious input. Input validation is the process whereby the input taken from the user is validated and ensures that the user did not input any strings of code that can bypass the security. If the code is not validated correctly, the hacker will be able to launch attacks such as SQL Injection and steal data from the database. In the rating field, users cannot submit anything that is more than 5 and less than 1. In the comments field, users cannot submit anything which has symbols such as ' , < > , / , @ , # , ! , = , - and Numerics because these symbols can be used as possible injection attack codes.

6.4 Secure Software Testing

6.4.1 Test Strategy

Testing strategy

After implementing all the features, it is important we check the usability of the feature to ensure the code achieve what they are supposed to do. The testing strategy employed is a combination of black box testing, white box testing, penetration testing and vulnerable testing.

- Black Box Testing

Black Box testing is a testing strategy that does not allow the tester to have any knowledge of the internal workings of the software presented. No inner workings of the software are revealed to the tester as this is to test the resiliency of the software, determining the outputs to the inputs of the tester as a form of behavioral analysis. This method can be used to identify and address security vulnerabilities proactively such that the risk of being attacked or hacked are minimized.

- White Box Testing

A White Box testing method is the opposite of black box testing. As the name suggest, it is where testers have full knowledge of the inner workings of the software designs and implementations. This is more commonly used to test use and misuse cases of the software. White box testing requires direct access to the source code to work, making it possible to detect embedded code issues such as spyware, backdoors, or logic bombs. The inputs are structurally analyzed to ensure that the code implementations follow a certain order and design specification.

6.4.2 Test Plan

Our testing plan consist of input validation test, cross site scripting test, SQL injection test, large file upload.

- Input Validation Test
 - Special characters such as “!”, “@”, “#”, “\$”, “%”, “^”, “&”, “*”, “(”, “)”, “<”, “>”, “?”, “/”, “\”, “|”, “[”, “]”, “{”, “}”, “.”, “,”, “`”, “~”, “-”, “_”, “+”, “=” are generally not allowed in all inputs unless specifically allowed (e.g. address input allows “#” alongside “.”, as some users would like to include the specific house number in their address details. A full stop is allowed since short forms such as street becomes “St.”, thus requires a full stop.)
 - When a user inputs an invalid input, it will display an error message beneath the input box but will show a specific message on what to input, and not show the entire regular expression so as to allow the attacker to brute force in an attempt to break the site.
- Injection Flaws Control Test:
 - Injection Flaws enable attackers to input malicious code through an application to another system. It can be though the use of external programs via shell commands and backend databases using SQL Injection.
 - Sources of input and the events where the software will be connected to the backend store or command environment. The sources can range from authentication forms, search input fields, or hidden fields in the webpages. When sources are determined, input validation tests are performed to ensure software is not susceptible to injection attacks.

- Scripting Attacks Controls Test:
 - The lack of output sanitization will cause scripting attacks. This test is performed to validate controls which mitigates scripting attacks. It ensures that the output is sanitized by escaping or encoding the input before it is sent to the client. Another one of the many tests to ensure is that scripts are unable to be injected into the input fields.
 - One way we tested on our own page was to log in as a seller and attempt to create a listing. Instead of listing an actual product, attempt to inject scripts into the webpage. The page will deny any scripting performed. If any scripts passed, check the source code to find that the scripts have all been sanitized before it was submitted into the database. The same can be performed for customer and owner input fields.

6.4.3 Test cases

6.5 Security Conclusion of the Feature

The review page was coded and tested to the best of our abilities to fulfil the CIA security traits. We have aimed to code it to prevent common attacks such as SQL injections, brute force and cross site scripting attacks. The review page may also be one of the first few entry points if the attacker has access to an account that can create or edit a review. We have tried and tested it vigorously to ensure it have no loopholes that could be exploited.