SER335 Engineering Secure Software Systems **Spring 2023 B Lab 3**

Objectives:

1. Create misuse cases to represent security requirements
2. Develop an understanding of how to use CAPEC and CWE national security tools
3. Apply basic Risk Management Framework modeling for actionable decision-making in a security context

***Submission**: Activity 1 may have one or more files, Activity 2 one file. Please prefix the Activity solution files with "activity1" or "activity2" so we can identify them. Put all files into one zipfile named ser335b_lab3_<asurtie>.zip*

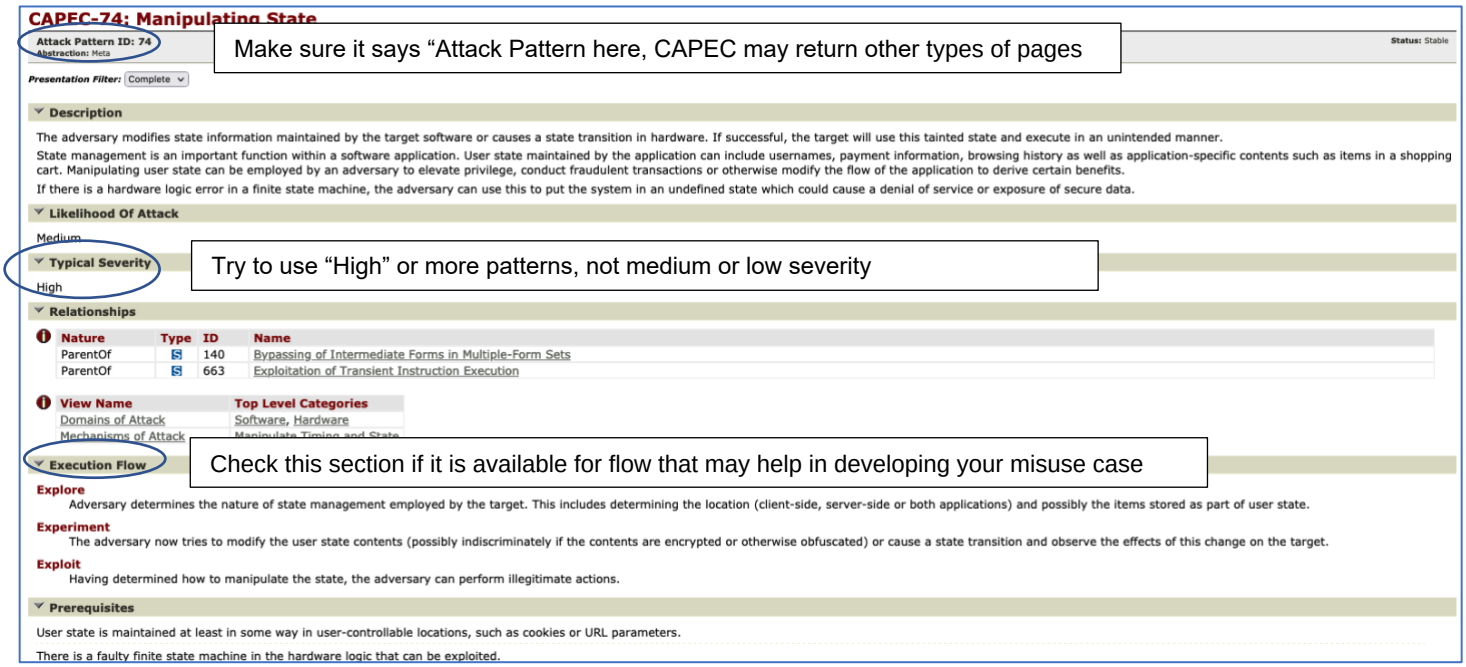
Activity 1: Misuses Cases (50 points)

You will follow the method below to develop misuse cases for a given web application based on CAPEC attack patterns. An *attack pattern* is like a *design pattern*; it is a repeatable type of attack (attempted exploit). CAPEC is a comprehensive dictionary and classification taxonomy of known attacks that can be used by analysts, developers, testers, and educators to advance community understanding and enhance defenses. Describe each misuse case using the given Misuse Case Template (see Appendix I). The requirements specification of the web application including use cases are provided to you (see Appendix II).

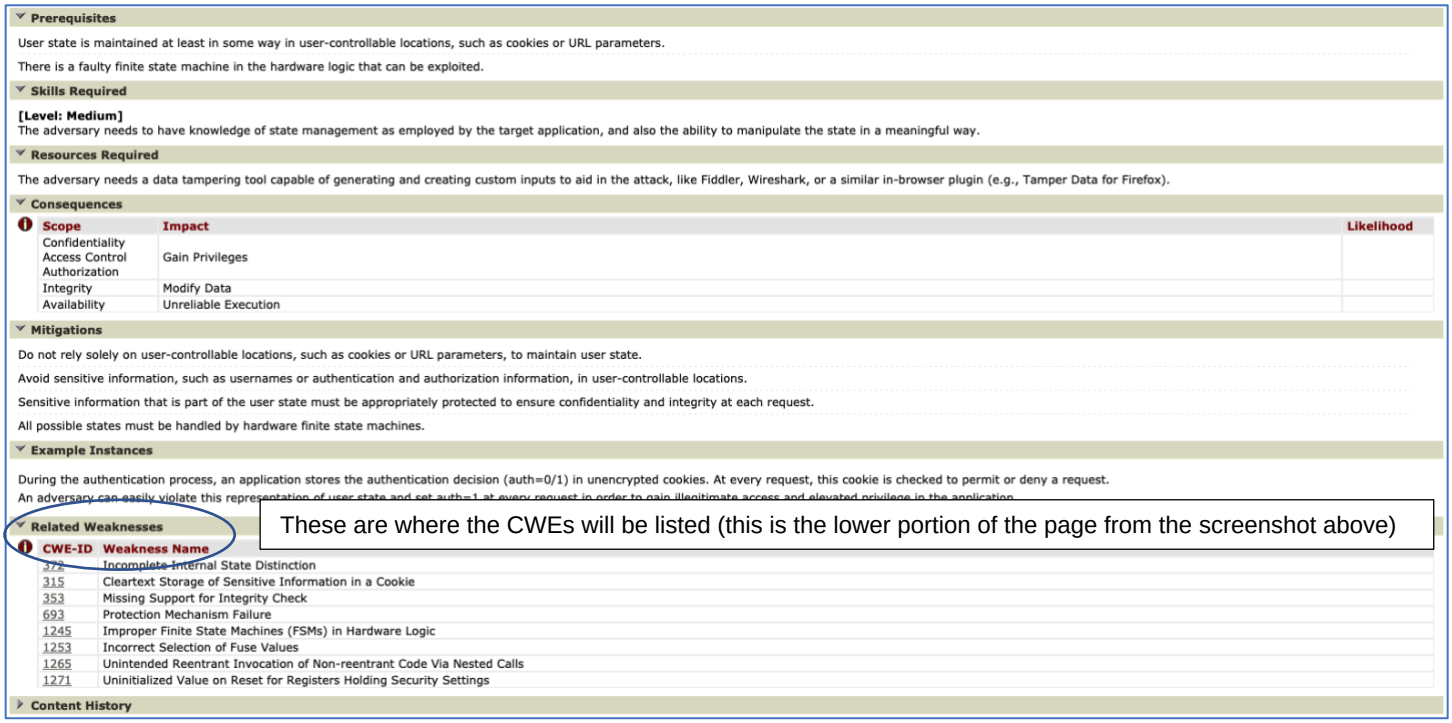
Method for Misuse Case Development:

1. For each use case, select words from the use case description to for searching attack patterns from CAPEC. Record these words. Using your keywords, use the CAPEC search tool at https://capec.mitre.org/. Just use the "Search CAPEC" bar on the left-side of this webpage. As shown in the capture below it will return links to CAPEC pages, most of which will be Attack Patterns.

2. From the List of Attack Patterns returned, select the most RELEVANT with highest SEVERITY as Misuse Case(s) (see screen capture below). To determine if an attack pattern is relevant to a use case, read the Description and the Execution Flow of the attack pattern. The flow should be applicable to the use case. Some attack patterns may not have a flow. In this case, you can write the flow of events based on the description/summary of the attack pattern (get creative). The screenshot below shows an Attack Pattern page for the "shopping" example given in this Lab task.



3. Use the information from the attack pattern to fill out the *Misuse Case Template (Appendix I)* for each regular use case (Appendix II). The Checkout use case is already provided for you as an example, complete the other 3 use cases. **Note:** If some of the fields/information are missing from an attack pattern, use your best judgement to fill out the *Misuse Case Template* and make a note of it in your misuse case. For example, if there is no flow in the attack pattern, you can make a note "missing attack flow from attack pattern" in the "Flow of Events" field of the misuse case and hypothesize your own set of steps.
4. The CWE is the *Common Weaknesses Enumeration* database from Mitre, the same folks who put out CAPEC (Mitre is a special contractor for the US gov). CWEs are basically *vulnerabilities* with corresponding *controls* (whereas attack patterns are common *exploits*). So basically, we want to reverse engineer from possible exploits back to root vulnerabilities and identify mitigating controls. In the detailed view of a CAPEC attack pattern, relevant entries from the CWE are linked. Select a CWE with the HIGHEST Likelihood of exploit from those returned. Using the mitigation information from the CWE, to add (what you think is) the best solution/mitigation for the misuse case.



Prerequisites

User state is maintained at least in some way in user-controllable locations, such as cookies or URL parameters.

There is a faulty finite state machine in the hardware logic that can be exploited.

Skills Required

[Level: Medium]
The adversary needs to have knowledge of state management as employed by the target application, and also the ability to manipulate the state in a meaningful way.

Resources Required

The adversary needs a data tampering tool capable of generating and creating custom inputs to aid in the attack, like Fiddler, Wireshark, or a similar in-browser plugin (e.g., Tamper Data for Firefox).

Consequences

Scope	Impact	Likelihood
Confidentiality Access Control Authorization	Gain Privileges	
Integrity	Modify Data	
Availability	Unreliable Execution	

Mitigations

Do not rely solely on user-controllable locations, such as cookies or URL parameters, to maintain user state.

Avoid sensitive information, such as usernames or authentication and authorization information, in user-controllable locations.

Sensitive information that is part of the user state must be appropriately protected to ensure confidentiality and integrity at each request.

All possible states must be handled by hardware finite state machines.

Example Instances

During the authentication process, an application stores the authentication decision (auth=0/1) in unencrypted cookies. At every request, this cookie is checked to permit or deny a request.

An adversary can easily violate this representation of user state and set auth=1 at every request in order to gain illegitimate access and elevated privilege in the application

Related Weaknesses

These are where the CWEs will be listed (this is the lower portion of the page from the screenshot above)

CWE-ID	Weakness Name
372	Incomplete Internal State Distinction
315	Cleartext Storage of Sensitive Information in a Cookie
353	Missing Support for Integrity Check
693	Protection Mechanism Failure
1245	Improper Finite State Machines (FSMs) in Hardware Logic
1253	Incorrect Selection of Fuse Values
1265	Unintended Reentrant Invocation of Non-reentrant Code Via Nested Calls
1271	Uninitialized Value on Reset for Registers Holding Security Settings

Content History

Submission:

- You must add FOUR misuse cases to the given model, not including the one given to you. *This means for the remaining 3 regular use cases, 1 of them should have 2 misuse cases.*
- For each misuse case, complete the Misuse case template (Step 3) for each. This includes recording the search keywords (Step 1), selecting the most relevant and severe (Step 2), and identifying the highest exploit likelihood. You may record these by doing screen captures of CAPEC similar to the 3 screen captures I show you above.
- Please update the use case diagram (shown at the end of this document). The Astah file is available on Canvas, or you can simply edit the image here and add the proper notation using an image editor or Powerpoint.
- Grading points distribution: each misuse case specification is worth 10 points, and 10 points for the updated use case diagram.

Activity 2: Risk Modeling

Background

Managing *risk* is a critically important part of an organization's approach to security. There are a variety of Risk Management Frameworks (RMFs) and methodologies used across organizations:

- The National Institute of Standards & Technology (NIST) RMF: https://csrc.nist.gov/Projects/risk-management
- The Fair Institute: https://www.fairinstitute.org/
- The OCTAVE model from SEI: https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=309051
- The Federal Financial Institutions Examination Council (FFIEC) Cybersecurity Assessment Tool: https://www.ffiec.gov/cyberassessmenttool.htm

These tools are "high-level" in the sense they address organizational concerns across the range of technologies the organization consumes, produces, maintains, and deploys.

At a lower-level, when addressing risk in specific products (software systems and applications), you rely on *threat modeling*. In Module 1 we discussed the terms *threat* (circumstances that have potential to cause loss or harm) and *risk* (likelihood that a threat agent will exploit a vulnerability). Threat modeling then is understanding what vulnerabilities may exist, how harmful it would be if exploited, and how likely they may be exploited. Once we have a threat model we can determine plans of action to *mitigate* risk or enact a *contingency plan* in the event of an actual *exploit*.

There are many methods that have been devised as decision-support tools to assist both high-level manager and low-level engineering teams decide where to expend resources to address risk. As it is impossible to eliminate security threats, it is therefore impossible to eliminate risk. The best we can do is understand it so we can devise strategies to minimize the impact of exploits.

Task: Apply a simple risk model

A simple model of risk is to compute the product of the *impact* (loss) of a threat, and the *likelihood* (probability):

Risk Exposure (RE) = Impact * Likelihood [RE = P(E) * L(E)]

Even for this simple quantifiable risk model, determining these 2 components is non-trivial. IMPACT: If an exploit occurs, how much harm will be caused, and how can we quantify it? The most direct way is *cost* ($$$), ranging from a worst case (enough to put us out of business) to a best case (some financial impact that is relatively small with respect to our organization's resources). LIKELIHOOD: this component is even tougher as it is hard to provide an accurate probability estimate in many cases, as it may depend on many factors or change over time. In this exercise you will be given the values to use in the equation, but as you go through this course and specific threats and associated defenses are discussed, I invite you to re-visit this simple RE equation and just think about how you might use something like this to determine the extent it is worthwhile to implement defensive measures in your code.

Part 1 (20 points): You are provided data on a hypothetical project in Table 1. This project has 3 threats (it is not important what those threats actually *are*). The team has identified 3 root causes, or underlying reasons why the threat may turn into an exploit. Some of these root causes apply to some of the threats, hence the 3x3 matrix of Table 1. With this data:

1. **Formulate RE value for each non-blank entry in Table 1. P(E) means the *probability* of an *exploit* (likelihood), while L(E) means the *loss* that occurs due to that threat becoming reality. You should provide 4 responses here. 8 points**
2. **Formulate the overall RE for a given Root Cause. You should provide 3 responses here.**
3. **Formulate the overall RE for a given Threat. You should provide 3 responses here.**

	Threats (assumed outcome of a threat model):	Threat 1:	Threat 2:	Threat 3:
	Root Cause	P(E) | L(E)	P(E) | L(E)	P(E) | L(E)
1	Inadequate understanding of security needs	20% | 25	30% | 10	
2	Poor understanding of needed technology	30% | 15		
3	Inadequate resources to complete			20% | 45

Table 1. P(E)|L(E). Example: read the entry in Row 1, Threat 1 as "there is a 20% chance that Threat 1 will result in an exploit due to the underlying root, and this will result in a loss of 25 business value points"

As a conclusion of Part 1, answer the following: Which combination of Root Cause / Threat is the riskiest and why? Which Root Cause by itself induces the most risk and why? Which threat induces the most risk and why?

Part 2 (20 points): What do you do about risk? How can you reduce your RE? Given that it is a product of 2 components, that product will go down if you minimize one or both components. When you minimize likelihood, you are doing a *preventative* exercise, since you are trying to reduce the probability of the exploit. This is called a *mitigation plan*. When you minimize the loss of an exploit, you are enacting a *contingency plan*. It would be great if we can have unbounded mitigation and contingency plans – meaning we can implement as many defensive measures as we want to reduce the probability a threat agent is successful, and we can also implement as many post-exploit recovery actions as possible so that we are not suffering undue harm from the exploit (for our organization or for our end users). But these plans take very real time and effort – it takes time to implement secure coding measures, secure quality

practices, secure testing practices, all in an effort to reduce probability by identifying and closing vulnerabilities. It takes time to prepare contingencies – compensating our users (for example, identity theft protection program), hiring lawyers, etc. Nothing is for free, these plans come with a real cost! The key difference is that a mitigation plan, when adopted, is always enacted; whereas a contingency plan is only enacted if the exploit occurs.

Tables 2 and 3 below contain mitigation and contingency data. Each table lists mitigation or contingency actions in the leftmost column, followed by a *cost* to implement that action. This is followed by columns for each of the threats from Table 1. The populated cells have a percentage. For Table 2, this is the extent to which the Risk Mitigation activity *reduces the likelihood of that threat.* For Table 3, this is the *likelihood that the contingency plan works.*

4. **Using Table 2, consider if a particular proactive *mitigation action* is worth doing across the entire project. 12 points**

For each risk, would you use a mitigation plan? A contingency plan? Both? Neither? Note that multiple requirements share the same risks. This means that a single mitigation or contingency plan may help you address more than one requirement.

Action	**Risk Mitigation – Risk Reduction/Cost**	Cost	RC1	RC2	RC3
1	Training	4		20%	
2	Add Resources	5		10%	10%
3	Prototype	3	10%		

Table 2. Risk Mitigation actions with threat reduction and cost values.

Let's do the first row as an example. A "Training" mitigation action has the effect of reducing the likelihood of Root Cause 2 – related threats by 20%. In Table 1, we have one threat (Threat 1) in the row for Root Cause 2, and it has a likelihood (P(E) value) of 30%. We reduce P(E) by 20%, or P(E) = 30% - 20% = 10%. This has the desired effect of reducing the overall RE by 20%. BUT, it comes at a cost of 4 business units – so did we actually benefit from enacting this mitigation action? When you work out the revised RE and factor in the cost, you would find the answer is *no*, because the cost outweighs the benefit (a 20% reduction of a 15-unit impact is less than a cost of 4 business units).

For #4 repeat this computation for rows 2 and 3 in Table 2, and after computing the revised RE and factoring cost make a recommendation as to whether the mitigation action should be enacted. Your answer should have 4 analyses:

a) Is it worth enacting the Risk Mitigation action "Add Resources" to address Root Cause 2?

b) Is it worth enacting the Risk Mitigation action "Add Resources" to address Root Cause 3?

c) Is it worth enacting the Risk Mitigation action "Prototype" to address Root Cause 1?

d) Is it worth enacting the Risk Mitigation action "Add Resources" given the combined impact on Root Causes 2 and 3?

5. **Using Table 3, compute whether a contingency plan is worthwhile for that Root Cause (across the entire project). 8 points**

Table 3 has percentages in each cell but these must be read differently. This is the *percentage likelihood that the continency plan recovers the loss incurred when a given Exploit happens*. This is why the columns say "Threat" and not "Root Cause".

Action	**Contingency Plan - Recovery Value/Cost**	Cost	Threat 1	Threat 2	Threat 3
1	Litigation	15	50%		
2	Purchase insurance in case we cannot resolve	12		60%	
3	Deploy a software patch to eliminate the vulnerability	10			70%

Table 3. Contingency Plan actions with cost and recovery values.

Let's do row 3 software patch as an example. You would read this as "If we have to deploy a software patch because Threat 3 becomes reality (is exploited), then it will cost us 10 business units and there is a 70% chance the patch is successful in recovering the business value". In Table 1 we see that Threat 3 has an L(E) of 45; if we deploy the patch, it will always cost us 10 units and it will expect to recover the 45-unit loss 75% of the time. Given that the cost is 10 and the recovery is then 70% * 45 = 31.5, *yes*, it is worth doing the contingency action.

For #5 repeat this analysis for rows 1 and 2. Row 1 will be more complex as Threat 1 has 2 values to consider in its column of Table 1. Your answer should have 2 analyses, one for each row.

Part 3 (10 points): Combine your analysis from Parts 1 and 2 to recommend a mitigation and contingency planning strategy. That is, what are the final set of combined mitigation and contingency actions you recommend putting place for your organization? To answer this question, you need to look at your mitigation and contingency analyses from Part 2, but now consider what it would mean to *combine them*. Is it worth having *both* a mitigation action and a contingency action that overlap? Just one? None? How do you balance the tradeoffs? The tricky part here is that Table 2 addresses *rows* of Table 1 while Table 3 addresses *columns*.

6. **One written answer is expected for Part 3 that explains your final strategy and uses the prior analyses to justify your recommendations.**

Appendix I. Misuse Case Template and Instruction

Associated Use Case/Functionality:

Keyword(s) used to search for the attack pattern:

Misuse Case Name:

Attack Pattern Name and ID:

Attacker:

Misuse Case Objective:

Precondition/Prerequisite:

Resources:

Flow of Events:

Post Condition:

Solution & Mitigation:

Misuse Case Description Template Instruction

Associated Use Case/Functionality: <*The use case or functionality that the misuse case affects>*

Keyword(s) used to search for the attack pattern: *<What was typed in to CAPEC>*

Misuse Case Name: *<The misuse case name should be based on the Use Case Name (from the Attacker perspective) and threats.* The misuse case should include attacker and/or user. For example, *instead of using "Steals motorcycle" as a misuse case name, it is suggested to use "Attacker steals owner motorcycle" as the misuse case name. >*

Attack Pattern Name and ID: *<The name of the attack pattern used for developing this misuse case along with its ID.>*

Attacker: *<A primary actor for the misuse case. Often this may be a "generic hacker", but sometimes this may be a specific person or role type depending on the misuse case. For example, some attackers (threat agents) may have to work within the organization being attacked. >*

Misuse Case Objective: *<What is the Attacker trying to achieve from this attack? Information from the summary of the attack pattern can be used in writing the objective. >*

Precondition/Prerequisite: *<The state that the system should be in for this abusive interaction.>*

Resources: *< The information or tools the Attacker should have for this attack to be carried out.>*

Flow of Events: *<How is the attack executed?> Should have the following components:*

a) Explore: How to check if the use case can be exploited
b) Experiment: Indication of possible vulnerability
c) Exploit: Execute a complete attack.

Post Condition: *<What would happen to the system if this attack is carried out.>*

Solution & Mitigation: *<How can the risk of this misuse case be minimized/removed?>*

Information from relevant CAPEC attack patterns can be used to fill out the above template. The following shows the mapping of the CAPEC attack pattern fields to the above template fields

Misuse Case Template Fields	CAPEC Attack Pattern Fields
Misuse Case Objective	Summary of attack pattern.
Precondition/Prerequisite	Attack Prerequisites
Resources	Resources Required
Flow of Events	Attack Execution Flow
Post Condition	Base this on the Summary of the attack pattern or any other information of the attack pattern.
Solution & Mitigation	Solutions and Mitigations of attack pattern, and the potential mitigation of the related CWE associated with the attack pattern.

Misuse Case Example (Note: The screen captures in the assignment specification show CAPEC for this case)

Associated Use Case/Functionality: Check Out

Keyword(s) that was used to search for the attack pattern: Shopping

Misuse Case Name: Attacker makes fraudulent transaction

Attack Pattern Name and ID: 74, Manipulating User State

Misuse Case Objective: The abusive interaction is to modify state information maintained by the Check Out process. State information, such as username, payment information, and other browser history that are used to complete a valid checkout transaction is manipulated by an adversary to conduct fraudulent transactions.

Precondition/Prerequisite: Knowledge of the application state conditions

Resources: No special resources required. An attacker can choose to use a data tampering tool to aid in the attack.

Flow of Events:

a. The customer clicks the check-out button.
b. The application displays the items in the shopping cart of the customer.
    i. The attacker can add or remove items during this step.
c. The attacker clicks the "continue to payment" button.
d. The attacker enters the payment information.
e. The attack then clicks the OK button to complete the order.
f. The application checks that the credit card is valid.
g. The attacker uses a tool to modify the information that will be sent to the server, for example, modify the price.
h. The application processes the transaction, and displays the confirmation page.
i. The application sends a confirmation via customer's email.

Post Condition: Fraudulent transaction is made. For example, the attack buys products with cheaper prices.

Solution & Mitigation:

a) Do not rely solely on user-controllable locations (cookies or URL parameters) to maintain user state.
b) Do not store sensitive information, such as usernames or authentication and authorization information, in user-controllable locations.
c) At all times, sensitive information that is part of the user state must be appropriately protected to ensure confidentiality and integrity at each request.

Appendix II. Software Requirements Specification (SRS)

Apparel and Footwear Online Shopping System Requirements Specification

Introduction:

This SRS describes functionalities of an online shopping system (Apparel/ Footwear).

Scope:

The intended audience is developers and security engineers. This SRS will be the basis for verifying and validating the system throughout the development stages.

Overview:

The Apparel and Footwear Online Shopping System allows users to buy apparel and footwear by choosing the listed items from the web application.

User Characteristics:

1. Administrator: User adds, deletes, and edits the apparels and footwear information in the online shopping system.
2. Customer: User purchasing items on the Online Shopping System.
3. Payment System: The system that accepts customer's credit card or the payment details and makes the charge.

Assumptions:

- Administrator and Customer login from the same interface. The administrator is redirected to the screen to manage the system while the customer is redirected to commercial screen of the application.
- Customers can browse the products without login.
- Checkout requires login to complete a purchase.

General Constraints:

1. The main constraint for the Apparel and Footwear online shopping system is to authenticate the customers who registered in the system.
2. The system could be accessed through any web browser which supports:
    - Front-end: JavaScript, CSS and HTML
    - Backend: PHP and SQL Database.

Functional Requirements:

1. The system supports customer registration (customers must be registered to purchase items.)
2. All the inputs should be validated. Invalid input data should be ignored and error messages should be given.
3. When a customer searches for a product, the search results must contain product info (material, price, available sizes, etc.), its stock status (in stock, out of stock), the estimated time for delivery, rating and customer reviews.
4. The system supports shopping cart (add/delete/modify cart content): a customer can buy several products in a shopping cart in one transaction.
5. When a customer checks out items in the cart, all mandatory fields should be filled by the customer.
6. Customers can pay with a credit card or a debit card.
7. The system must record all completed transactions, and email confirmations to customers.
8. Customers can check the status (e.g., received, processed, or shipped) of their orders from the website.
9. The Apparel and Footwear Online Shopping system shall send an e-mail confirmation to the customer that the items they ordered will be delivered to the shipping address along with order information.

Non-functional Requirements:

- The system must guarantee that costumers' sensitive data is transmitted securely.
- The system should be able to handle multiple transactions concurrently.
- The system must be available 24/7.
- The system must be easy to use and compatible with major web browsers.

Use-Cases:

1. Authenticate User: This login screen includes username and password fields, remember me button, and forgot username or password links. Additionally, it has a submit, cancel, and clear buttons.
   Main Success Scenario:
   a. Admin or Customer clicks on login.
   b. The system prompts the user for their account credentials.
   c. The Admin or Customer enters their username and password.
   d. The system authenticates the login request.
   e. The Admin or Customer gains access to the system functionality

2. Check out: Customer finalizes his/her purchases by checking out the items in the shopping cart. Payment is processed.
   Main Success Scenario:
   a. The customer clicks the check-out button on the web page.
   b. The application displays the items in the shopping cart of the customer.
   c. The customer can add or remove items, customer clicks the “continue to payment” button.
   d. The system displays the invoice page.
   e. The customer enters the credit card information or other payment information and clicks the OK button.
   f. The system checks that the payment information is valid.
   g. The system processes the transaction, and displays the confirmation page.
   h. The system sends a confirmation via customer’s email.

3. Update Customer's Account: User modifies his/her account profile.
   Main Success Scenario:
   a. Customer selects "Edit Profile"
   b. System displays categories of profile (personal, preferences).
   c. Customer selects category.
   d. System displays the detail of the specific profile.
   e. Customer updates detail, presses "OK".
   f. If information changed includes credit card information, system validates new information with external credit card system.
   g. The system prompts the customer to confirm.
   h. System updates customer profile.

4. Logout User: User logs out from the web application
   Main Success Scenario:
   a. The Admin or Customer clicks on the logout button
   b. The system logs the Admin or Customer out and invalidates the cookie/session
   c. The system redirects to the login page.

Use case diagram (with sample Checkout-related Misuse Case):