Assignment 1 - Development Team Project: Design Document

Team: The Red Team

Members: Liam Willson, Simon Bolder, Fergus Nugent

Word Count:

Introduction

Online retailers harbour large quantities of sensitive information that is of high value if a data breach occurs (Martin and Palmatier, 2020). This includes, most importantly, customer financial information, also personal information in user accounts and retail information. 98% of UK businesses have an online operation, 70% of shoppers buy goods online, and approximately 16% of UK retailers experience a cyber-attack or attempted attack everyday (NCRCG, 2023). This implies that at all times and at very frequent intervals, this information is at high risk of corruption. Even supermarkets are at risk, in 2021 SPAR & TESCO supermarkets were attacked online and customer's financial information was compromised in a large-scale cyber attack (NCRCG, 2023). As a knock-on effect, customer deliveries were delayed also. Thus, the importance and significance associated with ensuring that sensitive information is appropriately and sufficiently protected, is paramount. Therefore, this design document has been constructed to outline the 'how' the proposed application will be built and specify why certain design choices have been made to prioritise security, whilst simultaneously prioritising complete and efficient functionality.

Application Proposal

We propose a secure online retail application for a retailer of plant seeds and gardening accessories. In terms of the functional capabilities of the system, the application will allow **users** to create, delete and edit a secure account, and then create, delete and edit orders by accessing, searching and retrieving from a catalogue of items, and then process payment and input address and personal details to execute purchase. Also, the application will allow **administrator** intervention, including management functionalities and functionalities to maintain and update the system. Furthermore, the application will log and monitor the interactions of the **users** and **administrators** with the application, its database and thus, the sensitive information it harbours. Also, the development process will involve implementing application functionalities that are security risk mitigation methods, including: authentication/authorisation requirements and data encryption. These will be switch on- and off-able. Ultimately, the system will be prepared to particularly prevent 3 common types of attack: brute force attack (Luxemburk et al., 2021; OWASP, n.d.), denial of service attack (Humayun et al., 2020; OWASP, n.d.) and API injection (IEEE, 2022; OWASP, 2023; Salt, 2023).

As the application relates to non-functional capabilities, the building process will focus on achieving the 10 aspects of the final build as described in the ‘non-functional requirements’ section of this design document. These define the ‘nitty-gritty’ of the application specifications and thus, can be used to define what the system looks like from a tangible standpoint. They will be achieved via ensuring that the correct and most appropriate application development methodologies are utilised. Secure development methodologies will be utilised.

Application Resources/System Requirements

Hardware:

- ☐ Modern operating system: Windows 7 or higher, OR MacOS X 10.11 or higher

Software:

- ☐ Python programming language (ideally, latest supported version)
    - Libraries & Tools: pip, Flask, Fernet, pylint, flake8, mccabe, postman
- ☐ Javascript programming language (JSON)

Application Assumptions

- Interaction with the system can be malicious & data entered can be incorrect.
- Product information, pricing and availability will be consistent and maintained by administrators so database information remains accurate.
- User access of the system will be via a reliable internet connection.
- System implementation will abide by relevant legal and industry standards/regulations (GDPR - General Data Protection Regulation).
- The mitigations implemented in the development process to counter malicious use/attack will be sufficient.
- UML diagrams will be updated based on design and coding decisions.
- Python libraries may change or new ones be added based on design decisions.
- Design doc will be updated based on coding and design decisions.

Application System Architecture Requirements

Users & Their Roles:

There will be 2 roles that the program will allow access to and expect…

1. General Users (customers):
    ➔ Can access the frontend to perform purchasing, cancelling and editing of orders. And, creation, deletion and editing of a secure account.
2. Administrators:
    ➔ Can access the frontend and backend to perform system management activities. Inclusive of, database, API and logging and monitoring review.

System Front-End:

- ☐ Is a terminal/command line interface (CLI).
- ☐ Provides a simple UI for the user to interact with (appropriate to user role).
- ☐ Communicates with the user in a seamless, effective and efficient manner.
- ☐ Interacts with an API over HTTPS (Hypertext Transfer Protocol Secure) which retrieves data from the database/backend.
- ☐ Ensures a secure connection between database, backend and frontend.

System Back-End:

- ☐ Logs user interactions with the program and any access to its database.
- ☐ Includes an API that retrieves data from the database.
- ☐ Implements data encryption and sanitation.

- ☐ Implements input validation from input fields.
- ☐ Ensures a secure connection between database, backend and frontend.

System Database:

- ☐ Is intuitively and efficiently organised and easily manually updated by permitted users (administrators). Is written using JSON & its contents are API accessible.

Application Requirements

Functional Requirements:

- User authentication/authorisation.
- User account creation, deletion and editing.
- User creation, deletion and editing of orders.
- Logging and monitoring of user and administrator interactions.
- Payment processing (inputting of financial information and order execution).
- Providing a searchable product catalogue.
- Functionality to cart multiple products whilst shopping.
- Data encryption (of sensitive data).
- Provide a command line interface (CLI) UI.
- Provide switch onable/offable security measures.
- Develop a minimised attack surface.

Non-Functional Requirements:

Accessibility and Usability…

- The system terminology must be non-technical and easy for users to understand.
- The system must provide users with clear messages on errors.
- The system functionality must be user friendly.
- The system must process/search without case sensitivity.

Alerts, Logging and Monitoring…

- The system must provide a mechanism for logging and alerting.
- The system must log when an error occurs.
- The system must log when a user enters incorrect data.
- The system must create a system log that is only accessible by administrators.

Audit and Control…

- The system must have levels of access permissions that can be configured by an authorised person.
- The system must log an audit trail.
- The system must ensure data lineage is supported.

Backup…

- The system must have configuration to set backup intervals.
- The system must backup data.

Capacity…

- The system must have adequate storage to hold data and software applications.

Compliance…

- The system must comply with personal data processing (GDPR) rules.
- The system must meet 3-star Open Data (https://5stardata.info/en/) for logs.
- The system data must be machine-readable. Structured formats such as JSON.
- The system API must be compliant https://spec.openapis.org/oas/v3.1.0 .
- The system must prevent incorrect data entry.

Extensibility…

- The system must be extensible or require minimal re-writing of code if required using OOP standards.

Manageability…

- The system must have documentation supporting the use of the solution.

Security…

- The system must use appropriate encryption patterns for data/files.
- The system must use appropriate security for API connections.
- The system must mark user accounts that are inactive after a period of time.
- The system must allow administrators to have full control of user accounts.

Testing…

- The system must have appropriate supporting testing scripts.

Adherence to GDPR (General Data Protection Regulation) Standards

The GDPR Standards (General Data Regulation Protection, 2018) to abide by:

GDPR Standard	Our Adherence to the Standard
Lawfulness, fairness and transparency	The application will process the subject's data lawfully (UK law), fairly and transparently.
Purpose limitation	The application will process the subject's data for legitimate purposes; user accounts and ordering goods.
Data minimisation	The application will process only as much data as absolutely necessary for the purposes specified.
Accuracy	The application will collect accurate and up-to-date data.
Storage limitation	The application will only store personally identifying data for as long as required for the specified purpose.
Integrity and confidentiality	The application will ensure appropriate data processing security, integrity and confidentiality.
Accountability	The developers will ensure the application complies to GDPR.

Security Vulnerabilities/Risks & Mitigations

Analysis and consideration of the OWASP (The Open Worldwide Application Security Project, 2023) web application security risks, the following have been considered as most relevant to a compromise of the proposed system. Therefore, mitigations have been planned to maximise reliability, confidentiality and integrity of our proposed system:

YELLOW for most relevant risks.

OWASP Web Application Security Risk	**Proposed Mitigations**
A01:2021 - *Broken Access Control*	- Define user role (user, administrator) - Control access dependent on user role - Require authentication/authorisation - Alert administrators of suspicious access - Rate limit API
A03:2021 - Injection	- Using a secure API - Using parameterised queries - Using positive server-side input validation - Limit queries

OWASP Web Application Security Risk	**Proposed Mitigations**
A02:2021 - *Cryptographic Failures*	- Sensitive data encryption using secure protocol - Encryption is authenticated - Ensure cryptographic randomness to avoid predictable, low entropy encryption - Utilise Hypertext Transfer Protocol Secure (HTTPS) for web application data transfer - Store cryptographic keys securely as byte arrays - Secure server using Transport Layer Security (TLS)
A04:2021 - *Insecure Design*	- Utilise linter tools to rectify programming errors, bugs, stylistic errors/efficiency and suspicious constructs - Review of code by trusted 3rd parties - Utilise secure coding practices - Review via thorough functional testing
A06:2021 - *Vulnerable and Outdated Components*	- Use static and dynamic testing to update functionality - Utilise up-to-date software and versions - Ensure compatibility: tools, libraries and environments - Remove unused dependencies

OWASP Web Application Security Risk	Proposed Mitigations
A07:2021 - *Identification and Authentication Failures*	- Password creation, hashing and requirement - Harden API pathways - New hashes/keys and encryption occurs
A08:2021 - *Software and Data Integrity Failures*	- Perform frequent data backups - Implement a review process - Use of OWASP (2023) security supply chain tool
A09:2021 - *Security Logging and Monitoring Failures*	- Logging of auditable events; e.g. logs of applications application interactions and API - Ensure logging of financial transactions occurs - DevSecOps team has established alerting system such that suspicious activities are detected and responded to promptly - Plan for incident response and recovery of compromised data or case of malicious activity

Brute Force, Denial-of-Service and API Injection Attacks - Focus

Brute Force, Denial-of-Service and API Injection attacks are allowed by failures that relate to *A07*, *A04* and *A03*, respectively, in OWASP (2023). *A09* applies to all 3.

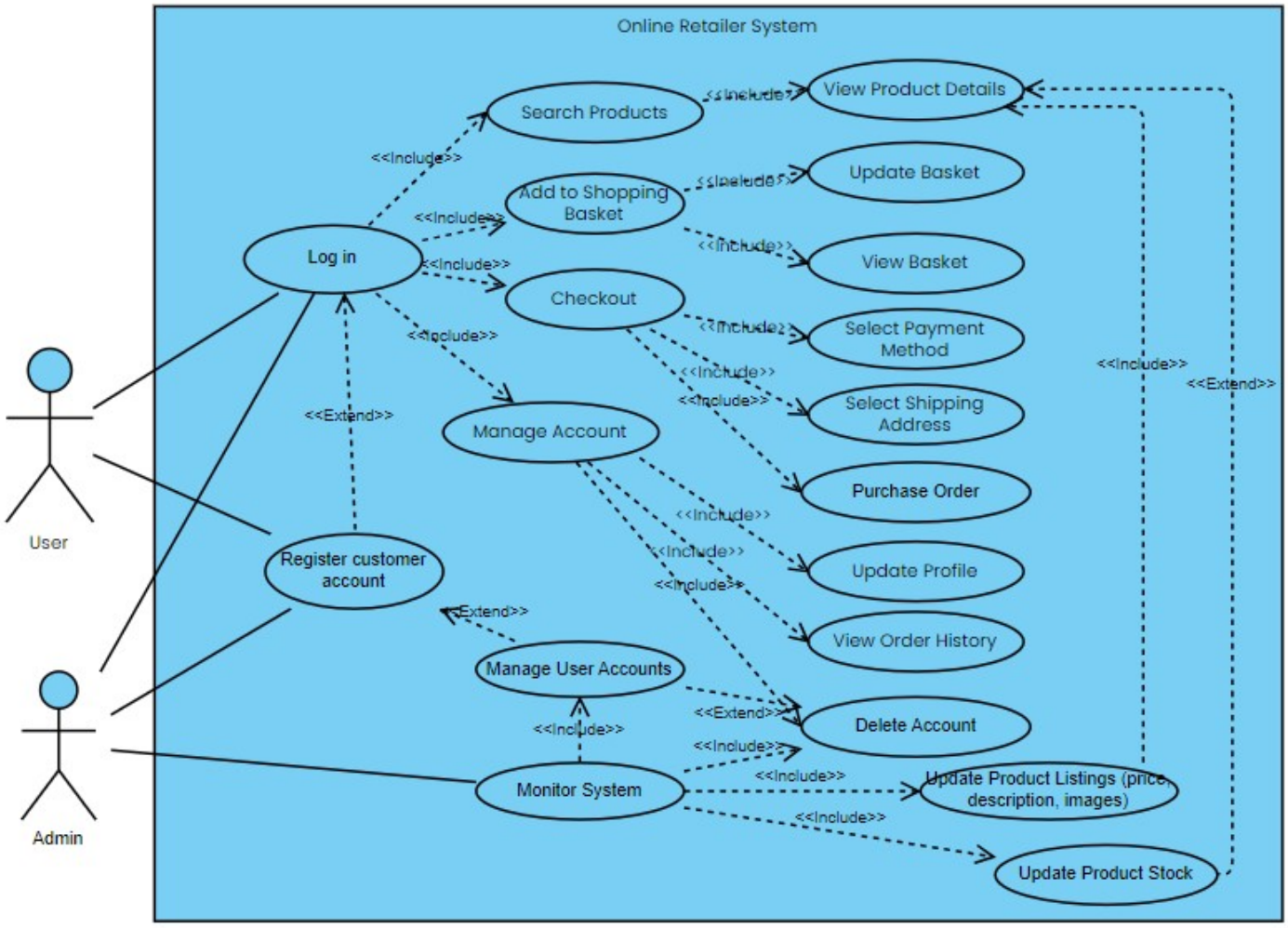
Application Tools, Libraries & Operating Environment

Web Framework	Flask 2.3.2 - A lightweight and flexible microframework that does not require particular tools or libraries.
Integrated Development Environment (IDE)	Visual Studio Code 1.81 - Code editor optimised for building and debugging modern web applications.
General Design Pattern	Model-view-controller (MVC) - General overview concept of the application development goal.
Database Management	JSON - To transmit data within the web application.
Python Libraries	pip, Flask, Fernet, pylint, flake8, mccabe
Python Testing	Unittest - Module for testing units of source code.
Linters	Pylint 2.17.5 - Code analyser used to identify programming errors. Flake8 6.1.0 - Code analyser used to check style compliance and syntactical errors. Mccabe 0.2 - Code analyser used to report cyclomatic complexity.
Logging & Monitoring	Using Python programming & JSON database

Testing Tools	pylint, flake8, mccabe
Security Modelling	OWASP (2023) - To consult and define security risks.
Development & Collaboration	Github - To build/store the application repository. Google Docs - To build the design documentation. Miro - Workspace to facilitate innovation and idea clustering. Trello - Dashboard to define and assign tasks for completion (scrum sprinting). Microsoft Teams - Platform for team meetings and face-to-face collaboration. Updates on the agile scrum process.
Development Process Modelling	Agile Scrum Methodology - Project management system defined by incremental development.
UML Creation Tools	Visual Paradigm 17.0 - Allows establishment of linkages between UML model components.

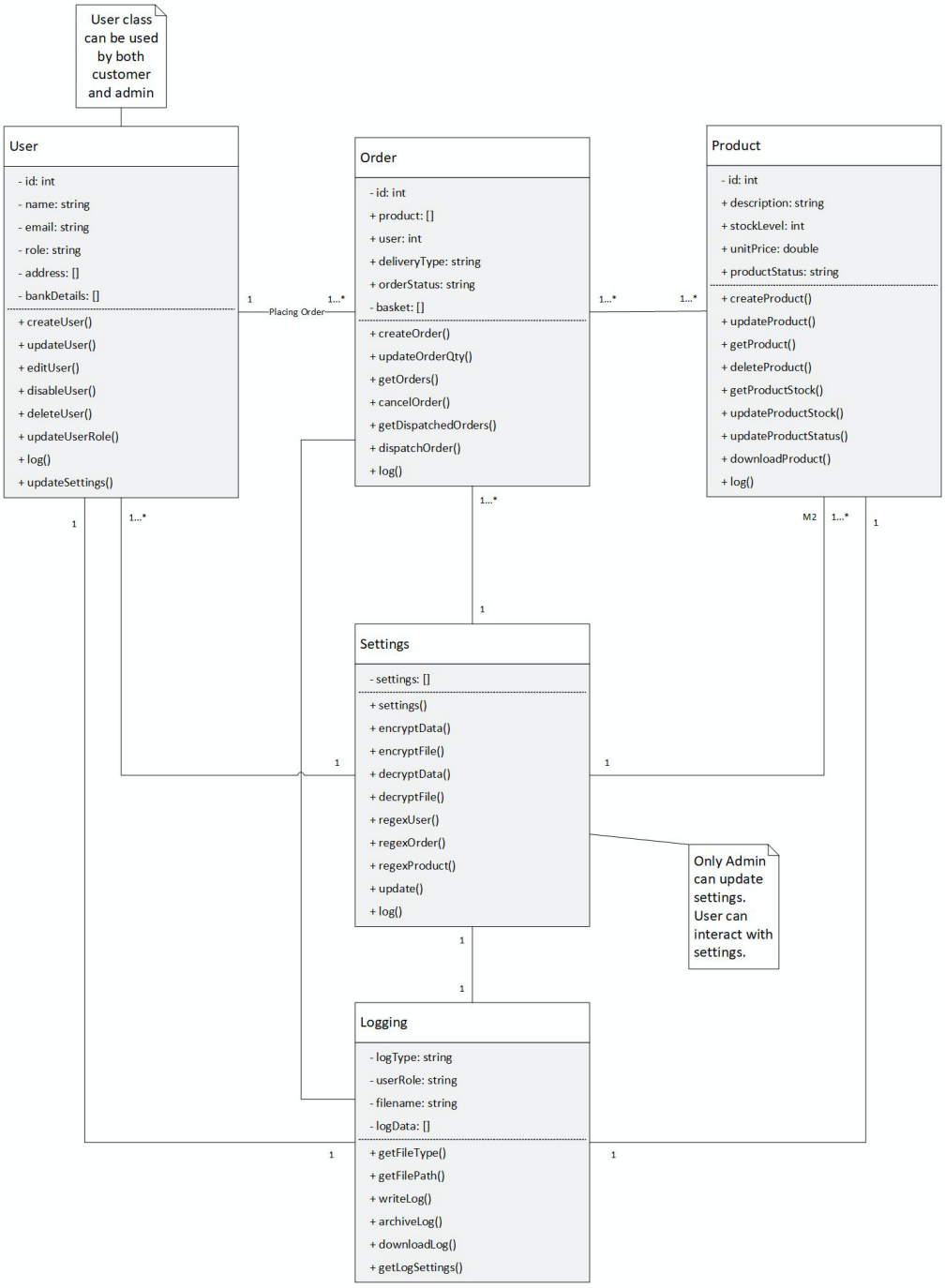
Appendix

Application Use Case Diagram:



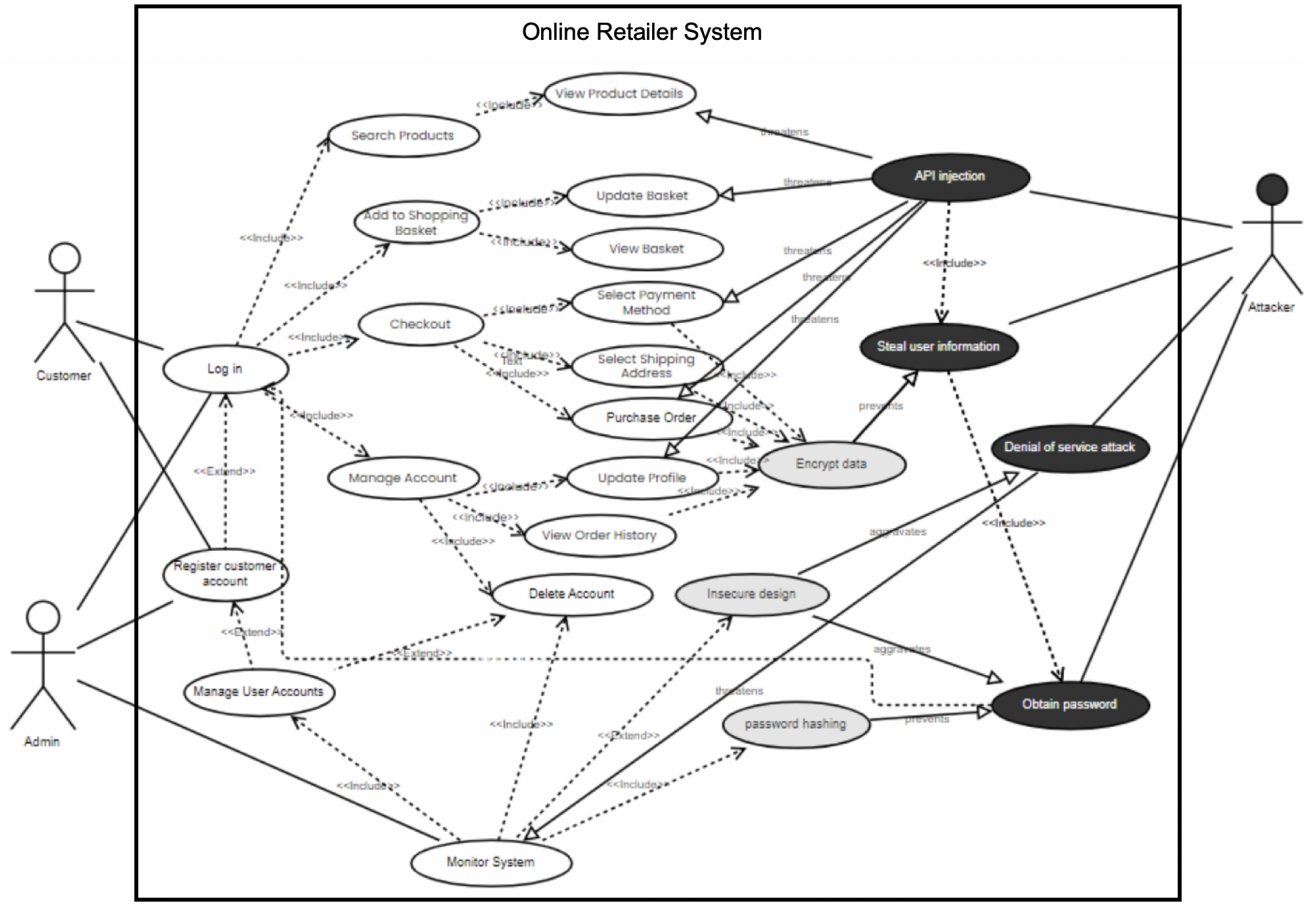
^^^ Fig. 1: This use case UML diagram Illustrates how the **user** and **administrator** will interact with the system and also how their interactions with the system will overlap and eventually interact. Particularly, it shows that only specific interactions and functionalities are available to each specific type of user. Furthermore, how the functionalities available to each type of user interact with downstream functionalities that the application can execute, is shown.

Application Class Diagram:



^^^ Fig. 2: This class UML diagram shows the structure of a system's class, attributes, operations and relationships amongst the objects. The dependency, associations, aggregations, compositions and inheritance within the system are represented here.

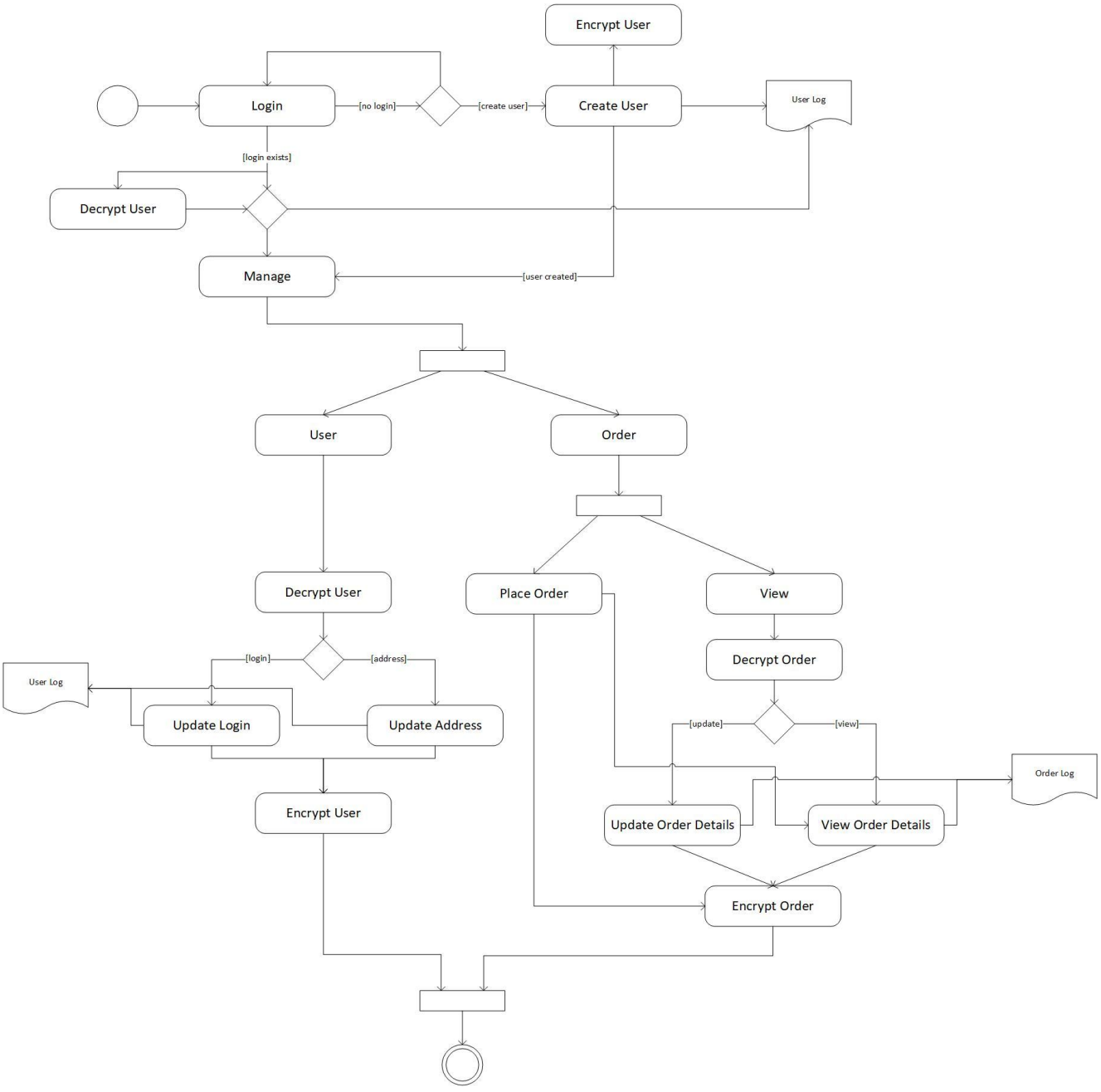
Application Misuse Case Diagram:



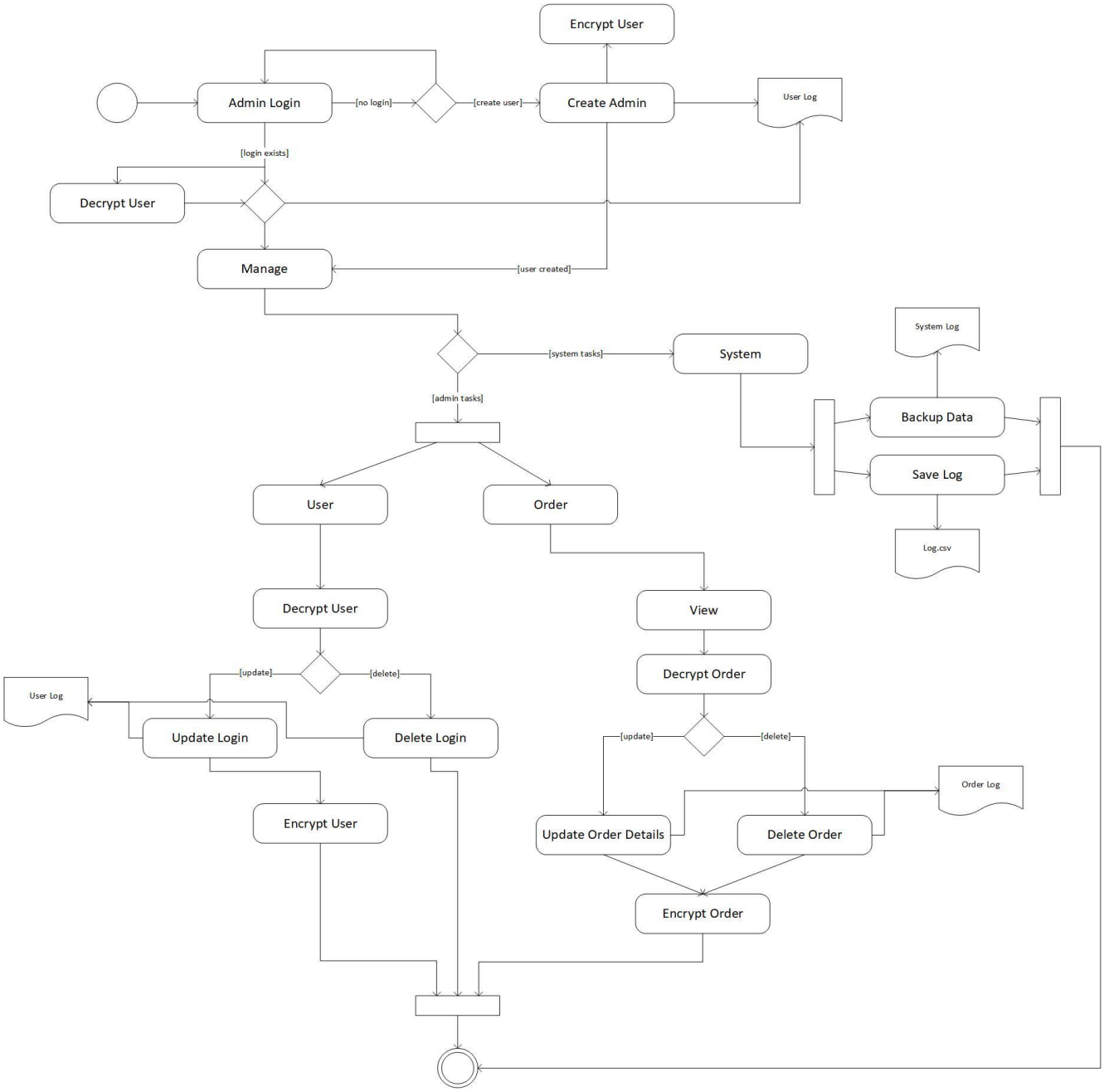
^^^ Fig. 3: This misuse case UML diagram shows the interaction of the **user** and **administrator** with the system, like the use case diagram. Although, importantly it also shows the methods by which a malicious **user** or **attacker** would compromise the application and it's data via brute force, denial-of-service and/or an API injection attack. Thus, the UML diagram focuses on presenting how these would occur and via which mechanisms or system weaknesses.

Application Activity Diagram:

for *Users*

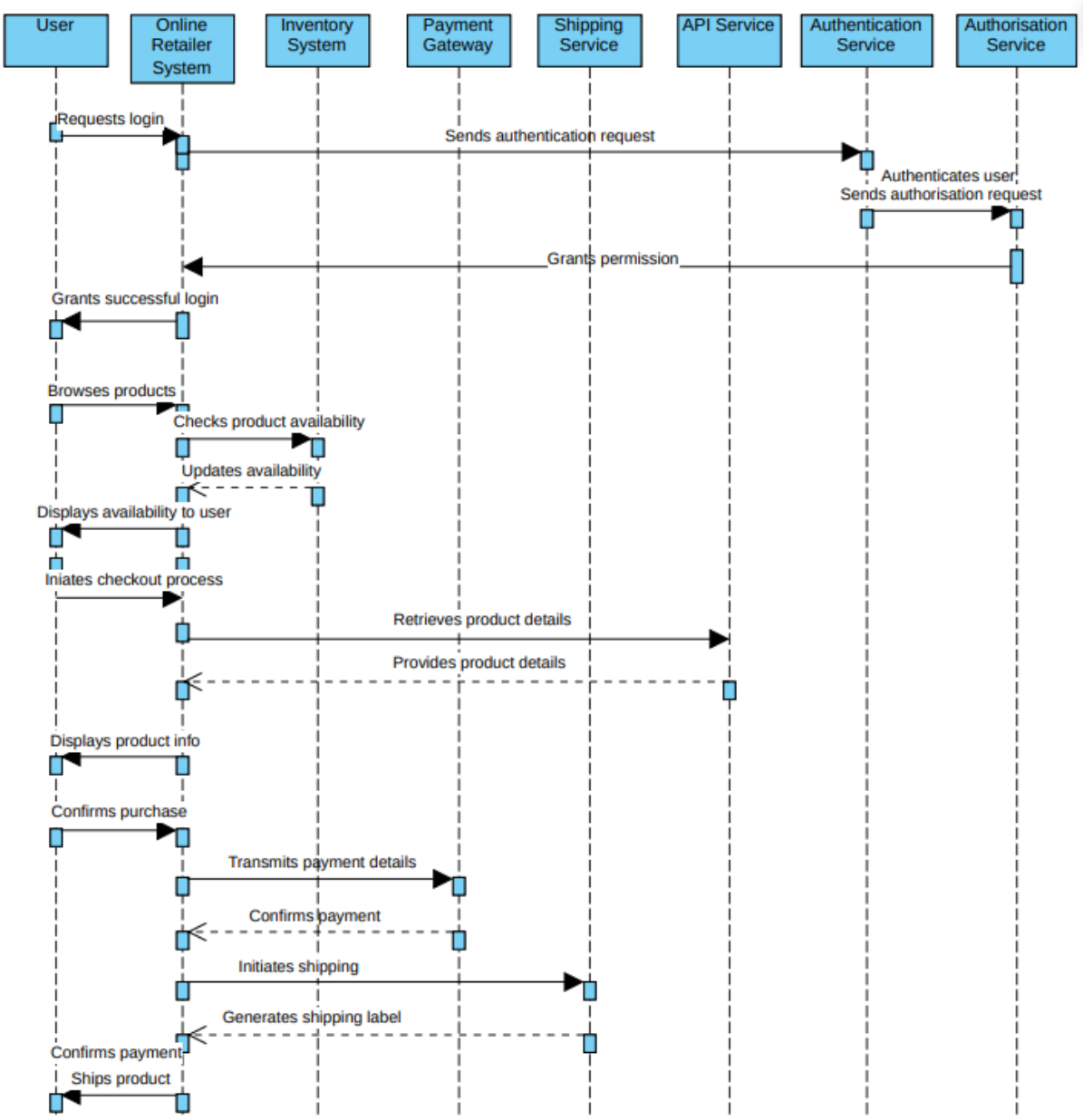


for *Administrators*



^^^ Fig. 4 & Fig. 5: These activity UML diagrams for **users** and **administrators** display the activities performed by the application system. The behaviour of the system is illustrated and so is the logic of the algorithm. The workflow process between **users** firstly and then **administrators** secondly, is displayed.

Application Sequence Diagram:



^^^ Fig. 6: This sequence UML diagram showcases how the application operations are carried out. Showing the interaction and relationships between the functionalities of the application in the context of a complete application overview. Specifically, it is a representation of the functionalities available to the **user** and the messages that can be immediate or downstream of the **user's** action.

Note: Data logging occurs at various stages of the online retail system for security, auditing and monitoring purposes, however, its operation in the background makes it not a necessary inclusion in the diagram.

References

General Data Protection Regulation (2023). General Data Protection Regulation Information. Available from: https://gdpr-info.eu/ [Accessed 6 Sep 2023].

Humayun, M., Niazi, M., Jhanjhi, N., Alshayeb, M. & Mahmood, S. (2020) Cyber security threats and vulnerabilities: a systematic mapping study. *Arabian Journal for Science and Engineering*, 45, pp. 3171-3189.

IEEE Computer Society. (2022) API Injection Attacks: What They Are and How to Prevent Them. Available from: https://www.computer.org/publications /tech-news/trends/api-injection-attacks-prevention [Accessed 7 Sep 2023].

Luxemburk, J., Hynek, K. & Cejka, T. (2021) Detection of https brute-force attacks with packet-level feature set. *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 114-122.

Martin, K. D. & Palmatier, R. W. (2020) Data privacy in retail: Navigating tensions and directing future research. *Journal of Retailing*, 96(4), pp. 449-457.

NCRCG. (2023) What Cyber Threats do Retailers face?. Available from https://www.nwcrc.co.uk/post/what-cyber-threats-do-retailers-face [Accessed 9 Sep 2023]

OWASP. (2023) OWASP Top Ten. Available from: https://owasp.org/www-project-top-ten/ [Accessed 22 Aug 2023].

OWASP. (n.d.) Blocking Brute Force Attacks. Available from: https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks [Accessed 7 Sep 2023].

OWASP. (n.d.) Denial of Service. Available from: https://owasp.org/www-community/attacks/Denial_of_Service [Accessed 7 Sep 2023].

Rumbaugh, J., Jacobson, I. & Booch, G. (1999) The Unified Modeling Language Reference Manual. Available from: https://idsi.md/files/file/referinte_utile_studenti/The%20Unified%20Modeling%20Language%20Reference%20Manual.pdf [Accessed 1 Sep 2023].

SALT. (2023) API Security Trends 2023. Available from: https://salt.security/api-security-trends?& [Accessed 7 Sep 2023].