Design and Development of an Automated and Secure Material Ordering System Aligned with NIST CSF 2.0 and OWASP Top Ten

BARIS YILMAZ, University of Twente, The Netherlands

This paper outlines the design and development of an Automated and Secure Material Ordering System (ASMOS). The system is intended for students and staff at the University of Twente as a tool to borrow hardware from the university for project work. The current manual ordering process requires multiple approval steps via email, often resulting in delays of over a week. These delays can hinder students' progress and negatively affect academic outcomes. Addressing this problem constitutes the primary motivation for this project. The proposed ASMOS is a full-stack web application designed to streamline the material ordering process by implementing multi-role workflows and automated approval chains, thereby reducing processing delays and manual interventions. ASMOS is built using Vite, React, and Bootstrap for the frontend, and Node.js with Express.js for the backend, which interfaces with a PostgreSQL database managed through Prisma. Security and data integrity are the main priorities. Key features include two-factor authentication, role-based access control, password hashing, and audit trails. This research explores established best practices for secure web application development through a review of relevant literature, with a particular focus on the National Institute of Standards and Technology Cybersecurity Framework (NIST CSF 2.0) and the Open Web Application Security Project (OWASP) Top Ten Web Application Security Risks. These guidelines form the foundation of ASMOS' security architecture. The system is evaluated based on metrics such as the reduction in processing time for key tasks (compared to the existing week-long email workflow), the minimisation of manual steps, and usability feedback gathered through semi-structured interviews with students and staff.

Additional Key Words and Phrases: data integrity, material ordering system, NIST CSF 2.0, OWASP Top Ten, secure web application, usability evaluation, workflow automation

1 INTRODUCTION

This section outlines the motivation and problem statement underpinning the project, introduces the proposed solution, and presents the research questions aimed at evaluating the system's efficiency in terms of security, user experience, and automation.

1.1 Motivation and Problem Statement

The University of Twente (UT) offers several modules in which students require materials for their project work. These materials are typically supplied by the university to the project groups. This research is motivated by such projects, where students may need to borrow specific materials from the university to complete their work.

A relevant example is the project offered in Module 5, 'Computer Systems', of the Technical Computer Science (TCS) programme at the UT. The current material ordering process in this module involves several manual steps. Students are required to email their supervising group teaching assistant (TA) with a request for the materials they need. The group TA then either provides feedback or approves the request. Approved requests are subsequently forwarded to the head TA, and finally to the teacher(s) for final approval.

This process can sometimes take more than a week, potentially hindering the progress of student projects and negatively affecting academic outcomes.

1.2 Proposed Solution

To address these inefficiencies, this project proposes the development of a new material ordering system, referred to as ASMOS: Automated and Secure Material Ordering System. ASMOS is intended to streamline the ordering and approval process and reduce the number of manual tasks involved.

The design and development of ASMOS are guided by two established cybersecurity resources: the Cybersecurity Framework (CSF) 2.0, developed by the National Institute of Standards and Technology (NIST) [6], and the Open Web Application Security Project (OWASP) Top Ten Web Application Security Risks [7]. These resources form the basis for the system's secure, role-based web application, which enables students, TAs, and teachers to interact with ASMOS in a controlled and reliable manner. The CSF provides a set of best practices and guidelines for developing, testing, and maintaining the cybersecurity of information systems, while the OWASP Top Ten identifies the most critical security risks facing modern web applications. Together, these frameworks provide the foundation for the security architecture of ASMOS. Further details concerning the NIST CSF 2.0 and OWASP Top Ten are provided in Section 2, *Background and Context: Cybersecurity Frameworks*.

1.3 Research Objectives and Questions

The objective of this research is to investigate how the efficiency of the current manual material ordering process can be improved by automating key aspects of the procedure, whilst maintaining data integrity and implementing robust cybersecurity measures. These measures include features such as two-factor authentication (2FA), access control, encryption, and audit trails.

Accordingly, the research questions are designed to evaluate the automation efficiency, system security, and user experience. The methodology for answering these questions is explained in Section 3.1, *Answering Research Questions*.

(RQ1) What cybersecurity best practices should be implemented to secure a multi-role academic material ordering web application such as ASMOS?

(RQ2) To what extent are students and staff satisfied with the usability of their respective interfaces in relation to the complexity of performing key tasks?

Author's address: Baris Yilmaz, b.yilmaz-1@student.utwente.nl, University of Twente, P.O. Box 217, Enschede, The Netherlands, 7500AE.

(RQ3) To what extent can the current material ordering process be streamlined and automated within ASMOS by minimising manual tasks, in comparison to the existing manual ordering process?

2 BACKGROUND AND CONTEXT: CYBERSECURITY FRAMEWORKS

As the system processes personal user data, including credentials, ensuring robust cybersecurity and maintaining data integrity are essential. For this reason, the NIST CSF 2.0 is adopted, and the OWASP Top Ten is utilised to identify security risks.

2.1 NIST CSF 2.0

NIST developed its CSF in consultation with hundreds of cybersecurity professionals [8]. It provides standardised best practices for securing digital systems. CSF 2.0 was published in 2024 as a follow-up to the decade-old CSF 1.0, which was originally intended solely for securing critical infrastructure, such as hospitals and traffic systems [8]. Earlier studies such as [2] and [5] have compared CSF 1.0 to other established frameworks, including those developed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). These studies demonstrated that CSF 1.0 effectively assisted cybersecurity professionals in securing critical infrastructure. However, these studies are now outdated as they emphasised that CSF 1.0 was not applicable to private (non-critical) organisations.

CSF 2.0 addresses this limitation by explicitly extending to non-critical organisations, including commercial companies and educational institutions such as the UT. CSF 2.0 is designed as a voluntary and flexible framework, intended to be tailored to the specific needs of the organisations implementing it[6].

The *CSF Core* is central to CSF 2.0. The CSF Core is structured around a set of fundamental cybersecurity functions that provide a high-level overview of an organisation's approach to managing cybersecurity risks. These functions are: *Identify*, *Protect*, *Detect*, *Respond*, and *Recover*. Together, they guide organisations through the processes of understanding their assets and risks, implementing safeguards, detecting incidents, responding appropriately, and restoring normal operations. In CSF 2.0, a sixth function, *Govern*, has been introduced, reflecting the need for accountability, risk management structures, and establishing clear cybersecurity policies [6]. These six functions support the development of tailored security practices.

In addition to expanding to non-critical organisations, CSF 2.0 simplifies the process for organisations to assess their cybersecurity with the introduction of *profiles*. The framework distinguishes between a 'current profile', representing the organisation's present cybersecurity maturity; a 'target profile', outlining the desired future state; and a 'profile gap', which identifies areas requiring improvement [6]. For further detail on CSF 2.0, the official NIST documentation [6] provides nuanced explanations that fall beyond the scope of this paper.

2.2 SURFaudit Standards Framework

The UT uses the SURFaudit Standards Framework [9], which is primarily based on ISO/IEC 27002, while also aligning with ISO/IEC 27001 for security management and ISO/IEC 27017 for cloud services [9]. Ideally, this research would adopt the same standards to ensure seamless integration with the university's information security policy. However, in this project NIST CSF 2.0 is selected over the SURFaudit Standards Framework, as it offers a modular and flexible approach enabling only the relevant security standards to be implemented in the ASMOS web application. In contrast, SURFaudit requires more comprehensive, organisation-wide compliance and formal certification processes, which are beyond the scope of this research project.

NIST CSF 2.0 is also voluntary and is designed to be accessible to development teams of all sizes. Although full adherence to all NIST standards is unlikely within the time frame of this module, its flexibility makes it more suitable within the scope and time constraints of this module. Nevertheless, the discussion of future work in Section 6.4 outlines how the SURFaudit Standards Framework could be integrated with the implemented NIST CSF 2.0 to ensure compliance with the UT's information security policy.

2.3 OWASP's Top Ten

NIST recommends that the CSF 2.0 be used in conjunction with other cybersecurity resources [6]. Consequently, the second major contributor to cybersecurity decisions in the research is the Open Worldwide Application Security Project (OWASP)'s Top Ten: The Ten Most Critical Web Application Security Risks [7].

Since 2003, OWASP has published six lists, identifying the most critical risks in web applications at the time of each respective release. The latest edition was published in 2021, with a seventh planned for 2025. These Top Ten lists are ideal to use in projects as they raise awareness, and can serve as a set of milestones during the development. For this research project only the 2021 version is considered, as the 2025 version is not yet released at the time of developing ASMOS, and the earlier versions are considered outdated. The OWASP Top Ten of 2021 can be read in Appendix D.

3 METHODOLOGY

The methodology comprises four main stages: a literature review, system design, system development, and an evaluation through usability testing.

The literature review focuses on identifying cybersecurity best practices relevant to ASMOS, drawing upon the NIST CSF 2.0 and OWASP's recommended prevention measures for the risks identified in the OWASP Top Ten. This literature review directly addresses research question RQ1. Only academic journal articles, peer-reviewed conference papers, and official documents from the UT, NIST, or OWASP are considered.

The second stage entails the system design, including the specification of requirements, the development of wireframes for the webpages, and the construction of the database schema. Following the design, the development phase begins. ASMOS employs Vite, React, and Bootstrap for the frontend, and Node.js with Express.js for the backend, which interfaces with a PostgreSQL database managed

through Prisma. The system design and development processes are discussed in detail under their respective sections.

The final stage of the methodology consists of usability testing with participants. Usability testing is accompanied by short, semi-structured interviews that help in answering the research questions as well as providing feedback on the user experience. Participants consist of TCS students, a TA from the Business Information Technology (BIT) programme, and a teacher responsible for managing the project in module 5 of TCS, which is the example project given in the introduction of the paper. Students were eligible to participate provided they had previously completed projects that required them to borrow materials from the university for their work; their academic grades for these projects were not part of the eligibility. Participants tested their respective webpages in ASMOS, and were also shown the remaining webpages to enable them to provide comprehensive feedback on the system as a whole.

3.1 Answering Research Questions

Each research question is addressed using a different method.

RQ1: Cybersecurity Best Practices.
RQ1 is primarily addressed through the literature review. Relevant publications on NIST CSF 2.0 [6], NIST's recommendations for password management [4], and the OWASP Top Ten [7] are analysed to establish a baseline for secure system design.

The answer to this question is used to evaluate the extent to which CSF 2.0 has been followed and applied within ASMOS. The OWASP Top Ten then serves as a checklist of risks that have been mitigated. Finally, this provides a foundation for the discussion of future work, including potential improvements to the prototype to enhance compliance with both NIST and the SURFaudit Standards Framework adopted by the University of Twente.

RQ2: User Experience Satisfaction.
RQ2 is addressed through a combination of literature review, usability testing, and semi-structured interviews with participants.

During the interviews, participants were asked to provide feedback on potential improvements to the system based on their experiences. They were specifically asked for their opinions regarding the complexity of the web application and whether they would prefer stricter security features (such as strong password requirements and enforced 2FA), even if these measures reduced ease of use when performing key tasks, such as requesting materials (as students) or approving orders (as TAs and teachers).

RQ3: Improved Automation.
In contrast to RQ1 and RQ2, RQ3 is addressed through empirical comparison. This entails a direct comparison between the current method of material ordering and the new ASMOS web application. Relevant metrics include, for instance, the time taken from the initial order request to the point at which students obtain the materials, as well as the reduction in the number of manual interventions required when using ASMOS instead of the existing email-based ordering process.

A limitation to this empirical comparison is that the number of participants is substantially smaller compared to its real-world usage in modules, where several project groups may want to place orders in a short time frame. This can affect the time required for teachers to distribute materials, as they may become overwhelmed by numerous incoming requests that take additional time to prepare and to set up time slots with students to collect the items. Alternatively, insufficient stock availability may cause additional delays that this research cannot anticipate, as no materials are being ordered from webshops in the user testing.

4 DESIGN PHASE

The design phase comprises the creation of wireframes for the webpages, the specification of requirements, and the design of the database schema.

4.1 Requirement Specifications

The following list presents the non-functional security requirements for ASMOS. These requirements were primarily formulated based on official documentations provided by NIST [6] [4] and OWASP [7]. Only non-functional security requirements are included in this section, as these are the focus of this paper.

NIST refers to conventional passwords as '(memorised) secrets'; to keep it consistent with other sources, only the term 'password' is used throughout these requirements and the rest of the paper.

The CIA triad —Confidentiality, Integrity, and Availability— indicates which pillar of cybersecurity each requirement addresses. Appropriate citations are included for the requirements that were taken from outside sources. Requirements are prioritised according to the MoSCoW method: Must haves, Should haves, Could haves, and Will not haves.

- R1 Login sessions **MUST** be secured using JSON Web Tokens (JWTs), specifically access tokens and refresh tokens [7]. *(Confidentiality & Integrity)*
- R2 As ASMOS uses role-based access control, the system **MUST** be protected against unauthorised access through URL modification, ensuring users cannot access pages beyond their permitted roles [7]. *(Confidentiality)*
- R3 JWTs **MUST** be stored securely using HTTP-only cookies and the web application's context memory. Session tokens **MUST NOT** be stored in plain cookies that are vulnerable to Cross-Site Scripting (XSS) attacks. *(Confidentiality & Integrity)*
- R4 ASMOS **MUST** employ an approved, encrypted, and authenticated communication channel (e.g., HTTPS) when requesting passwords, in order to provide resistance to eavesdropping and Man-in-the-Middle (MitM) attacks [4]. *(Confidentiality & Integrity)*
- R5 Sensitive configuration variables **MUST** be stored in environment (.env) files on the server rather than hardcoded in the source code. *(Confidentiality)*
- R6 Upon account registration, the password **MUST** be checked against a known list of commonly used, expected, or compromised passwords. This list **COULD** include passwords obtained from previous breaches, dictionary words, repetitive or sequential characters (e.g., 'aaaaaa', '1234abcd') and context-specific words, such as the name of the service, the username, and derivatives thereof. Passwords found on this list **MUST** be rejected. The system **MUST** inform the user

of the reason and provide guidance for selecting a stronger alternative [4]. *(Confidentiality)*

R7 The system **MUST** have a rate-limiting mechanism for limiting failed attempts [4]. *(Confidentiality)*

R8 Passwords **MUST** be stored in a form that is resistant to offline attacks. Passwords **MUST** be salted and hashed using a suitable one-way key derivation function. Key derivation functions take a password, a salt, and a cost factor as inputs then generate a password hash [4]. *(Confidentiality)*

R9 The system **MUST** make use of Cross-Site Request Forgery (CSRF) tokens that prevent CSRF attacks. *(Confidentiality & Integrity)*

R10 Data page checksums **MUST** be enabled at the time of database cluster creation to detect data corruption. *(Integrity)*

R11 The system **MUST** be protected against Distributed Denial of Service (DDoS) attacks. *(Availability)*

R12 The system **SHOULD** enforce the setup of 2FA during account creation, which **SHOULD** be required at each login. *(Confidentiality)*

R13 The system **SHOULD NOT** allow copying text from the password field. *(Confidentiality)*

The following three requirements are not directly tied to any of the CIA triad, but contribute to the usability of creating secure passwords which is indirectly tied to Confidentiality.

R14 The system **SHOULD** facilitate the use of password managers by allowing "paste" function in password fields [4].

R15 The system **SHOULD** offer guidance to the user, such as a password strength meter [4].

R16 In order to assist the user in successfully entering a password, the system **SHOULD** offer an option to display the password — rather than a series of dots or asterisks — until it is entered [4].

There is a mismatch between the UT's minimum password requirements [10] and NIST's recommended password requirements [4]. ASMOS adheres to the password requirements specified by the UT, because this is the intended organisation the system is tailored to. This is a benefit of NIST's flexible approach. These requirements are as follows:

R17 Passwords **MUST** be between 14 and 60 characters in length. (NIST recommends a maximum of at least 64 characters, but the UT enforces a 60 character maximum due to technical limitations.)

R18 Passwords **MUST** contain at least one lowercase and uppercase letter.

R19 Passwords **MUST** contain at least one digit.

R20 Passwords **MUST** consist solely of ASCII printable characters. The UT does not require special characters, but does not allow characters that are not native to a standard English International QWERTY keyboard.

R21 Passwords **MUST NOT** contain spaces. NIST recommends allowing spaces, the UT does not allow these.

Finally, as previously mentioned, the OWASP Top Ten 2021 serves as a checklist of the most critical risks, which ASMOS **MUST** be protected against.

4.2 Webpage Descriptions and Wireframes

Students, group TAs, head TAs, and teachers each have access to different webpages within ASMOS. This subsection presents wireframes for the most functionally significant webpages that are used to perform key tasks. Trivial and administrative webpages such as those for logging in, creating an account, adding items to the inventory, and user management are included in Appendix C for completeness and to conserve space within the main body of the paper.

Student-Facing Pages.
The student pages, as shown in Figures 1 and 2, allow students to browse the inventory, select items and specify quantities. Students are also required to provide a motivation for why they need the requested materials.

Borrowing materials is generally free of charge. However, if the university does not have a requested item in stock, the group members can still request it via ASMOS to inform the teachers what items to order from external webshops. Each group has an allowance, which limits the number of out-of-stock items the group can request and is only reduced when requesting such items. This is reflected in Figure 1, where two example items are in the inventory: one in stock and available free of charge, and another currently out of stock, where an associated cost per piece is displayed. The total cost of the order, as shown in Figure 2, only sums the costs of the out-of-stock items.

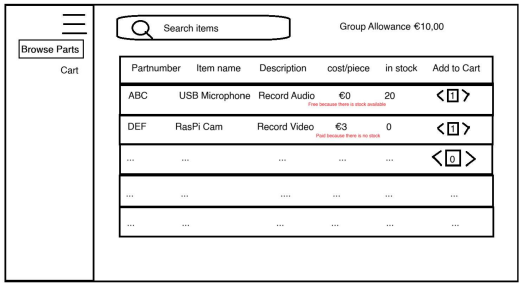

Fig. 1. Student Inventory Page Wireframe

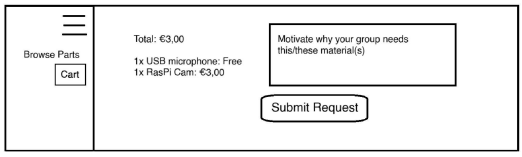

Fig. 2. Student Cart Page Wireframe

Order Management Page.
After an order is submitted, it becomes visible to the group TAs, head TAs, and teachers on the order management webpage, shown in Figure 3. This webpage implements the approval workflow described in the introduction. Group TAs are responsible for approving or rejecting the new orders from the group they supervise. Once group TAs have approved orders from their group, head TAs review all

incoming orders next, and either approve or deny them. Finally, the teachers review the orders and schedule a pickup moment with the group members to collect the requested material(s).

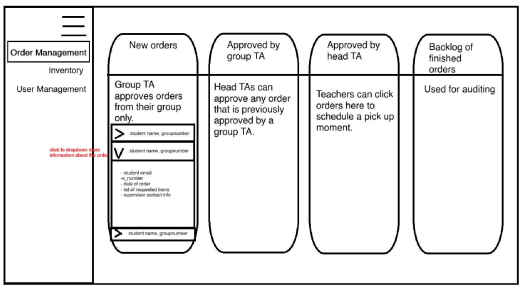

Fig. 3. Order Management Page Wireframe

This design is inspired by Trello's column-based layout, which is widely used for visual task management (Figure 4). In particular, the familiar "To Do", "Doing", and "Done" columns are adapted to represent the different stages of order approval within ASMOS. This visual structure provides an intuitive overview of pending, in-progress, and completed orders.

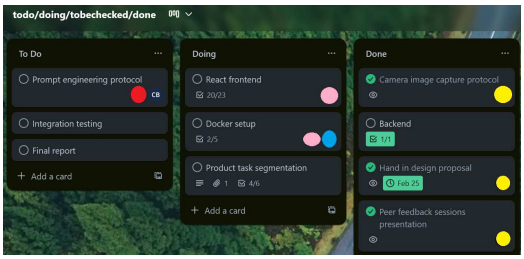

Fig. 4. Trello-style Columns

4.3 Initial Database Schema

The database is hosted locally and implemented using PostgreSQL due to prior experience with the language. Prisma, an Object-Relational Mapper (ORM), is used in the back-end to interface between the application and the database. Figure 5 shows an automatically generated database schema in Visual Studio Code.

The bottom left table `public:_prisma_migrations` is generated by Prisma for migration purposes, so it does not contain data regarding ASMOS. Prisma additionally has the benefit that it uses parametrised queries that prevent SQL injections (SQLi), which is the third most critical risk in OWASP's Top Ten 2021 (See Appendix D), alongside other injection-based attacks, such as XSS, which are addressed separately.

5 DEVELOPMENT AND FINAL IMPLEMENTATION

This section outlines the modifications made to the database compared to the version created during the initial phase. It also presents the finalised webpages that were developed in this project and discusses the implemented security measures, supported by test data.

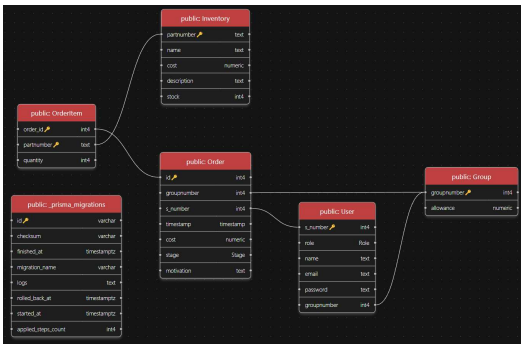

Fig. 5. Initial Database Schema

This section outlines the modifications made to the database compared to the version designed during the initial phase. It also presents the finalised webpages that were developed and discusses the implemented security measures, supported by test data.

5.1 Final Database Schema

The finalised database schema, shown in Figure 6 contains several modifications. The first is the addition of a boolean `deleted` in the `Inventory` table. This enables teachers to soft delete items by setting this field to `true`, which hides the item from student-facing pages without introducing foreign key conflicts that can occur if that item is referenced in `OrderItem` table for an existing order.

The second notable change is the addition of the table dedicated to storing refresh tokens. This table stores the refresh tokens alongside the `ipAddress` and `userAgent` for each login session for auditing suspicious activity. The table has a composite primary key consisting of the refresh token itself and the unique identifier for the corresponding user. This design permits users to log in multiple browsers or devices simultaneously, thereby improving usability.

The second significant change is the introduction of a dedicated table for storing refresh tokens. In addition to the tokens themselves, this table stores the `ipAddress` and `userAgent` of each login session, enabling the auditing of suspicious activity. The table uses a composite primary key comprising the refresh token and the unique identifier of the corresponding user. This design permits users to log in from multiple browsers or devices simultaneously, thereby improving usability.

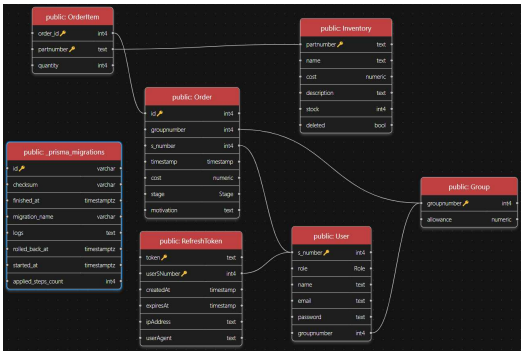

Fig. 6. Final Database Schema

5.2 Finalised Webpages

The student home page, shown in Figure 7, combines the functionality of the two separate pages designed during the initial phase. Once an item is added to the cart, the '+Add' button is replaced by three new buttons to decrement, increment, or remove all items of a particular type from the cart. The cart icon only becomes visible once at least one item has been added. Upon clicking the icon, the student is presented with the side menu shown in Figure 8 with an overview of all items in the cart and a field to enter their motivation for the requested items. This implementation is inspired by this tutorial for a React shopping cart [11].

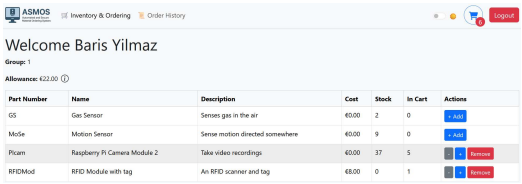

Fig. 7. Student Home Page Displaying the Inventory

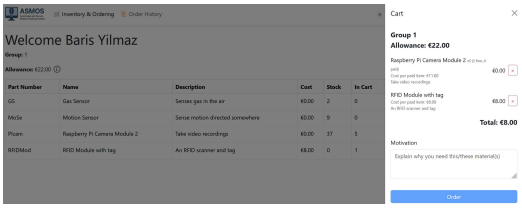

Fig. 8. Cart Displaying All Items

In addition, students can view the order history of all members of their group on a second page, shown in Figure 9. This functionality was not part of the initial design phase, as it was introduced at a later stage of development.

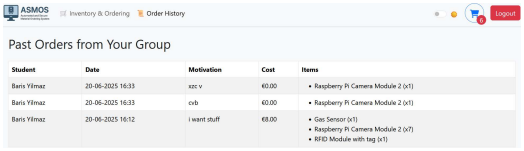

Fig. 9. A Group's Order History

Group TAs, head TAs, and teachers all have access to the pages shown in Figure 10 and 11, but each have different available tasks.

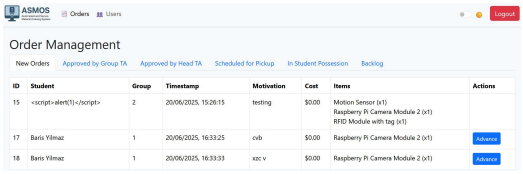

Fig. 10. Order Management Page: New Orders

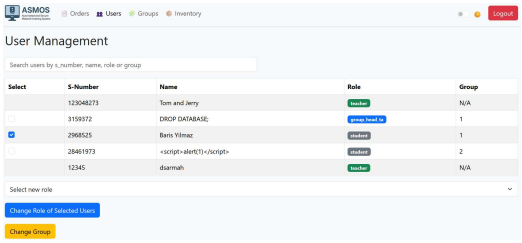

Fig. 11. User Management

Starting with the Order Management page: Group TAs (including head TAs who also supervise a group) initiate the process by approving the orders originating from their own group. After the group TA approval, the head TAs can approve the orders from any group. Subsequently, the teacher can schedule a pickup moment with the students for them to obtain the ordered material. They may schedule an single order, or select multiple orders to notify all relevant students via email.

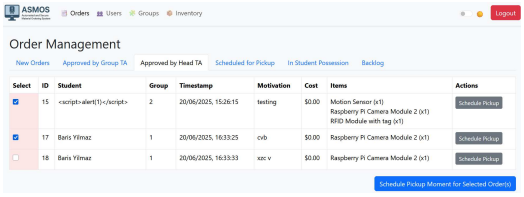

Fig. 12. Teacher Can Schedule Orders

This sends an email to all group members associated with the selected order(s). This feature utilises the free trial of Twilio SendGrid, as that lies within the scope of this research at no cost. UT's own `no-reply@utwente.nl` may be a better alternative in future development.

Fig. 13. Email Notification

5.3 Implemented Security Features

Strong Password Requirements with Feedback to Users.
The passwords used within ASMOS adhere to the same requirements as specified by the UT [10]. Although these are stricter than NIST recommends[4], it is an obvious choice to copy the UT requirements

as the intended users of ASMOS are also students and staff of the UT.

Hashed Passwords.
The passwords are hashed with bcrypt using a cost factor of 12, something that is considered a good balance between security and processing speed with modern computer hardware. As this study shows, a password with *just* 7 characters (and cost factor 12) was still unable to be brute forced after five days of trying [3].

JWTs: Access Tokens and Refresh Tokens.
As shown previously the database now has a field for refresh tokens. These are used in conjunction with access tokens that are sent to the frontend in HTTP-only cookies that are invulnerable to XSS attacks, so that users can stay authenticated without having to log in constantly; adding onto a user friendly design, whilst ensuring that security is maintained. Figure 14 shows this security working as intended: no tokens can be gathered via JavaScript code.

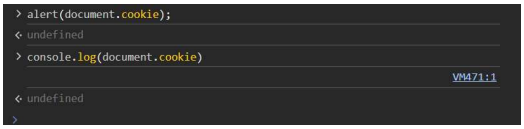

Fig. 14. Testing for Cookies

Protected Routes.
If a user is trying to visit a page that they do not have access to within their role, they will be redirected to the page shown in Figure 15.

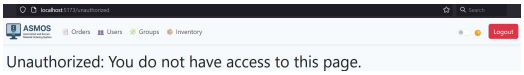

Fig. 15. Unauthorised page: Teacher Attempting to Access `/student-home`

The backend routes are protected as well: API requests must originate from authorised roles, this prevents situations where students may send a request to add an item to the database (which only teachers are allowed to do).

However, the prototype of ASMOS uses HTTP instead of HTTPS and runs locally on the laptop used for development. In production, the last step for protected routes would be to use HTTPS so that requests to and from the server containing sensitive information, including credentials and JWTs, are encrypted during transmission.

Injection Prevention.
Fields that allow plain text user input are sanitised. This was verified by creating test accounts with names such as "DROP DATABASE;" and "<script>alert(1)</script>", neither of which resulted in harmful code execution.

6 RESULTS AND DISCUSSION

This section provides the answers to the research questions, discusses the answers and lays out what can be worked on in the future for the ASMOS application.

6.1 Answering RQ1: What cybersecurity best practices should be implemented to secure a multi-role academic material ordering web application such as ASMOS?

Through literature review of [6], [4], and [7], the requirements have been specified. These requirements form ASMOS' *starting profile*. Ultimately, the *target profile* is to be on par with UT's cybersecurity, which can be achieved by aligning with the SurfAudit Standards Framework. Because this is out of the scope of this research and the prototype, only the aforementioned security requirements will be reflected upon.

The requirements R1, R2, R3, R5, R8, R10, R13, R14, R15, R16, and the password requirements R17 - R21 have been implemented successfully. The remaining requirements are either not possible within the scope of the research, or was not tested properly yet. However, this does not mean that the other requirements are less important. These requirements show areas of improvement within ASMOS, which is part of the *profile gap* as explained in CSF 2.0. The requirements that have been implemented to this stage, and the *Identification* (a function of CSF Core) of other security threats show that ASMOS is heading in the right direction.

For instance R4, using HTTPS for an encrypted communication channel is not feasible for the research module, because this would have to be paid for. R11, protecting against DDoS, depends on the server ASMOS is hosted on, in this case a laptop used for development. It is not possible to test this requirement in its development phase without dedicated server hardware. R6, cross checking passwords against known or easy passwords, and R7, implementing a rate-limiting mechanism: these were unable to be finished in the time constraint of this module. However, as later described in Section 6.4, having UT's Microsoft accounts connect to ASMOS with Surfconext may make R6 not applicable any more because ASMOS would not facilitate account creation at that stage. Similarly, R12 will be addressed as UT accounts have 2FA by default. Then finally R9, mitigating CSRF attacks and from OWASP Top Ten 'Server-Side Request Forgery' are not implemented and are for future work to reach the target profile.

6.2 Answering RQ2: To what extent are students and staff satisfied with the usability of their respective interfaces in relation to the complexity of performing key tasks?

This question is answered with a thematic analysis of the interviews and the hands-on user testing of the web application.

Creating an Account.
During the user testing and the interview, participants expressed difficulties with the registration form that is required to create an account. Particularly, participants struggled with the password requirements and ensuring the "confirm password" field matches the initial password. On top of that, the button to view the password is a 'view on hold' mechanism, whereas they preferred a 'view on toggle' approach; both are commonly used in online systems, such as the Windows lock screen that shows password on holding the eye icon only, and Google uses a checkbox that toggles the visibility of

the password. Although most participants understood the reasoning for the strict password, the combination of not being able to view both password fields at the same time with a difficult-to-remember password, made the registration process more difficult.

Several students and a teacher pointed out it would be a good idea to have an account confirmation email sent to the used email address upon registration, as users may mistype their s-number or email address. This is an excellent suggestion, because the prototype of ASMOS is a full-stack web application that is not connected to the UT's Microsoft accounts. Ideally ASMOS would be connected to the UT system to eliminate the need for account creation altogether, which will be covered more in Section 6.4.

Student Pages.
During the interviews, student participants were asked what they thought of the complexity of the student pages and ordering materials from the interface. The general consensus was that the pages are easy to use and intuitive. An annoyance two students noticed was that the group allowance does not always refresh on the page, which is an unintended bug in this prototype, but still something that affects the user experience.

6.3 Answering RQ3: To what extent can the current material ordering process be streamlined and automated within ASMOS by minimising manual tasks, in comparison to the existing manual ordering process?

The current process primarily consists of communication via email. Students also have no insight in the available stock at the time of ordering, so TAs and teachers have to consider who ordered first to fairly distribute the materials to the groups that ordered it when there was still stock remaining, and which groups were too late and have to wait for new stock to arrive. With ASMOS, if an item is out of stock, it informs the students by dynamically updating the student home pages to show the items must be ordered and group allowance will be used. This saves time going back and forth between TAs and students when TAs later realise an order cannot be fulfilled within the group allowance.

TAs and teachers also see the timestamps when all orders were placed in a concise table with the oldest order that should then have priority at the top, this saves time that would otherwise be spent on looking at the dates and times the emails were sent to each individual group TA's inbox.

Teachers have an easy method of adding, deleting, and modifying items to the inventory. However, it is their responsibility to keep this table up-to-date, for example, the costs of items can change after it was added to the database, ASMOS does not automatically update prices based on real world changes in the webshops. Another aspect: if, for example, a sensor is found to be defective, then teachers must manually decrease the available stock by one to account for this defective sensor.

Scheduling pickup moments for students to grab their ordered items from the teachers has also been improved. A teacher can click the checkboxes next to previously approved orders to send emails to all students in those groups with the same pick up moment, which may also save time compared to looking up the email addresses of students.

6.4 Limitations and Future Work

ASMOS is built with the limitation that users cannot simply log in with their existing UT accounts, and no groups are fetched from Canvas - where the groups already exist. This would make the management of users and groups centralised to Canvas, as there would not be a need to specify roles and groups again in ASMOS. Rather it can gather this information from Canvas. That would leave ASMOS to focus on its main functional requirements, such as ordering material, approving the orders, and controlling the allowance and the items in the inventory. A registration form would then also not be necessary, but a login form will be used for the users to log in with their UT account.

A similar full-stack web application that integrates with UT accounts is Grasple, which is used by the UT and other universities to provide math homework to students. Universities can connect their domain to Grasple with SURFconext, which is part of the benefits when complying to SURFaudit Standards Framework and receiving a certificate. This opens yet another opportunity for ASMOS: allowing a student to be in more groups at a time for multiple modules, making ASMOS a university-wide tool, rather than in a single module.

In terms of legality, there is also the General Data Protection Regulation (GDPR) [1] that ASMOS must comply to before it can be deployed in a production environment. For example, users must be able to request, rectify, or delete all the stored information, among other requirements to lawfully comply to this regulation.

Teachers still have some manual tasks surrounding the inventory management. These manual tasks do not necessarily reduce the automation compared to the existing ordering system, but these are areas of improvement ASMOS can aim to facilitate in in the future. Most notably in the prices that can change daily, API calls for items on various webshops can help in automatically adjusting the price of the items in real-time. Teachers would only have to input the links of such items once for setting up the inventory, afterwards the prices could dynamically change.

7 CONCLUSION

The current implementation of ASMOS is a prototype; by definition it is incomplete. There are still necessary steps to be taken to improve upon the security, primarily being able to use existing UT Microsoft accounts and connecting with Canvas groups system, and complying with the GDPR and SURFaudit Standards Framework. Aside from these, there is also feedback from the user testing to further refine the application, with the general consensus from students that they had an easy time performing key tasks, whereas the teachers may still benefit from using API connections to webshops to maintain accurate pricing. All in all, this prototype lays a foundation of how the University of Twente can automate the borrowing process in a secure manner.

REFERENCES

[1] 2016. General Data Protection Regulation (GDPR) – Legal Text. https://gdprâĂŚinfo.eu/. https://gdprâĂŚinfo.eu/ Regulation (EU) 2016/679. Official

Journal L119, 4May2016; applicable from25May2018.
[2] Riza Azmi, William Tibben, and Khin Than Win and. 2018. Review of cybersecurity frameworks: context and shared concepts. *Journal of Cyber Policy* 3, 2 (2018), 258–283. https://doi.org/10.1080/23738871.2018.1520271 arXiv:https://doi.org/10.1080/23738871.2018.1520271
[3] Toras Pangidoan Batubara, Syahril Efendi, and Erna Budhiarti Nababan. 2021. Analysis performance BCRYPT algorithm to improve password security from Brute Force. *Journal of Physics: Conference Series* 1811, 1 (Mar 2021), 012129. https://doi.org/10.1088/1742-6596/1811/1/012129
[4] Paul A. Grassi, Michael E. Garcia, and James L. Fenton. 2017. *Digital Identity Guidelines: Authentication and Lifecycle Management.* Technical Report Special Publication 800-63B. National Institute of Standards and Technology (NIST). https://pages.nist.gov/800-63-3/sp800-63b.html Accessed: 2025-06-11.
[5] Ahmed Ibrahim, Craig Valli, Ian McAteer, and Junaid Chaudhry. 2018. A security review of local government using NIST CSF: a case study. *J. Supercomput.* 74, 10 (Oct. 2018), 5171–5186. https://doi.org/10.1007/s11227-018-2479-2
[6] National Institute of Standards and Technology. 2024. *Cybersecurity Framework 2.0.* Technical Report NIST CSWP 29. U.S. Department of Commerce. https://doi.org/10.6028/NIST.CSWP.29
[7] OWASP Foundation. 2021. OWASP Top 10 – 2021: The Ten Most Critical Web Application Security Risks. https://owasp.org/Top10/ Accessed: 2025-05-23.
[8] Lei Shen. 2014. The NIST Cybersecurity Framework: Overview and Potential Impacts. *The SciTech Lawyer* 10, 4 (2014), 16–19. https://www.americanbar.org/content/dam/aba/publications/scitech_lawyer/2014/summer/nist_cybersecurity_framework_overview_potential_impacts.authcheckdam.pdf
[9] University of Twente. 2020. Policy on Information Security. https://www.utwente.nl/en/cyber-safety/cybersafety/legislation/policy-on-information-security.pdf.
[10] University of Twente. 2025. UT account (incl. activation, password change). https://www.utwente.nl/en/service-portal/workplace-support/accounts-passwords/ut-account-incl.-activation-password-change
[11] Web Dev Simplified. 2022. How To Create An Advanced Shopping Cart With React and TypeScript. YouTube. https://www.youtube.com/watch?v=lATafp15HWA

8 APPENDICES

A USE OF AI STATEMENT

During the preparation of the web application I have used ChatGPT as a search engine to learn React from (as this is my first time using React), some of the backend configurations, and style/bug fixes. After using ChatGPT, I have used other tutorials and templates that I have edited to suit my needs for the web application. I do not have insight on how **all** the other programmers have programmed these templates (whether by hand or with AI), however, some of these tutorials were videos on YouTube where it was programmed by the YouTubers themselves.

The logo of ASMOS has been AI generated and is used with permission of my supervisor.

As already disclaimed in the body of the paper, the database schemas are generated automatically in Visual Studio Code using Microsoft's PostgreSQL extension based on the tables in the database.

Typing checker tools (Overleaf's built-in spelling checker, Grammarly, and Word's built-in checker) have been used to check for spelling, grammar, and other typing issues, and ensured consistent use of academic British English language, rather than unintentionally combining American and British English throughout the paper. No full bodies of text have been blindly copied from any of these tools without reviewing them; I have gone through each individual suggestion and changed the paper accordingly if that particular suggestion made sense in the context, other changes I did not agree with were left out.

B GITLAB LINK

The GitLab repository can be found here: https://gitlab.utwente.nl/s2968525/researc project.

C WIREFRAMES OF NON-VITAL WEBPAGES

C.1 Login and Registration

Figure 16 shows the login and registration pages. These pages are considered straightforward. However NIST's recommendations for secure authentication are taken into account in the wireframe design: Clicking the eye-icon reveals the password and a password strength meter is included to encourage strong passwords.

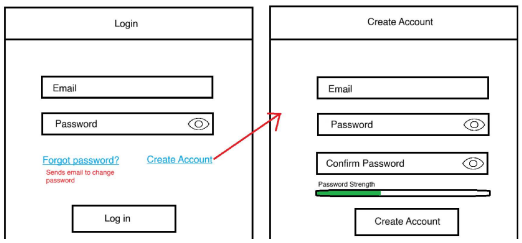

Fig. 16. Login and Registration Page Wireframes

C.2 User Management

This webpage (Figure 17) allows teachers to change the role of other users. In the design phase, the wireframe did not include the ability to assign a student to a group. This limitation was addressed in the final implementation of the user management webpage.

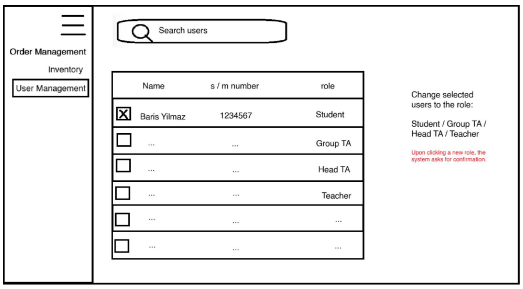

Fig. 17. User Management Page Wireframe

C.3 Inventory

Teachers can use the webpage in Figure 18 to add items to the inventory, which are then displayed on the student-facing pages. Teachers can either add one individual items manually, or upload a `.csv` or `.xlsx` file to add multiple items in a single request to the backend.

D OWASP TOP TEN 2021

Table 1 shows the categories of the identified risks in OWASP Top Ten 2021 with a description and examples for each category.

Table 1. OWASP Top 10 Security Risks (2021 Edition)

Category	Description	Examples
Broken Access Control	Failures that allow unauthorised access to systems, data, or functionality.	Bypassing login screens, viewing unauthorised pages via URL modification, tampering with API requests or JSON Web Tokens (JWTs).
Cryptographic Failures	Weak or missing encryption exposing sensitive data.	Transmitting data in plain text, storing passwords without encryption, using outdated cryptographic algorithms, relying on HTTP instead of HTTPS.
Injection	Insecure handling of user input leading to code or command execution.	Unsanitised inputs, SQL Injection (SQLi), Cross-Site Scripting (XSS).
Insecure Design	Inherently insecure features or workflows due to flawed system design.	Business logic that bypasses security checks, insecure default workflows, fundamental design weaknesses.
Security Misconfiguration	Incorrect or missing security settings in applications or infrastructure.	Revealing stack traces in error messages, unused default accounts, insecure server configurations.
Vulnerable and Outdated Components	Use of outdated libraries, frameworks, or software with known vulnerabilities.	Unpatched operating systems, outdated third-party libraries, known vulnerable software components.
Identification and Authentication Failures	Weaknesses in verifying user identity or enforcing authentication mechanisms.	Weak passwords accepted, ineffective password recovery processes, missing or broken multi-factor authentication (MFA), passwords stored in plain text.
Software and Data Integrity Failures	Lack of mechanisms to verify software or data integrity, leading to tampering or compromise.	Insecure software updates, unsigned code, lack of integrity checks for critical files or configurations.
Security Logging and Monitoring Failures	Insufficient detection, logging, or response to security events.	Missing logs, failure to monitor suspicious activities, inability to detect or escalate security breaches.
Server-Side Request Forgery (SSRF)	Exploitation of server functionality to make unauthorised internal or external requests.	User-supplied URLs fetching unauthorised resources, accessing internal systems via SSRF vulnerabilities.

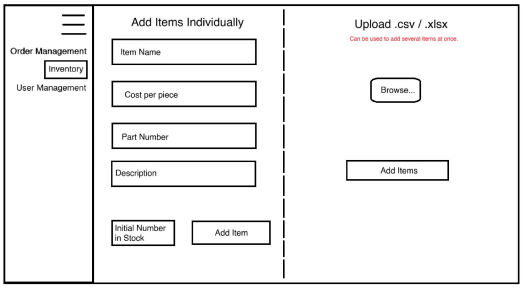

Fig. 18. Inventory Management Page Wireframe