1st December, 2023

COM2008

SYSTEMS DESIGN AND SECURITY

GROUP ASSIGNMENT : FINAL REPORT

Team 47

Team Members

Muhammad Ahsan Adnan
Joel Foster
Lekha Mohta
Muskan Sharma

Index

<u>Introduction</u>

This project aims to develop a comprehensive software system for Trains of Sheffield, a model railway retailer, introducing an innovative in-store computer system to efficiently manage product, customer, and order information. Trains of Sheffield specialises in selling a variety of model railway products, available in boxed sets or as individual models/parts, encompassing locomotives, rolling stock, track packs, and controllers, each with distinct product codes and retail prices.

The Trains of Sheffield system caters to different stakeholders, including customers, shop staff members, and a manager. A Customer can seamlessly browse, order, and purchase products by providing valid bank details. Staff members, who also function as customers, possess additional privileges to add, view, and update product data, as well as manage orders. However, they are restricted from accessing customers' banking details to ensure privacy and security. The manager, inherently a staff member, holds the authority to oversee a comprehensive list of users and can appoint or dismiss staff members. The managerial role is pre-assigned in the database to ensure a structured hierarchy within the system. This initiative seeks to enhance Trains of Sheffield's operational efficiency, customer experience, and overall management capabilities through a tailored and user-friendly software solution.

Team Strategies and Approaches

Throughout the project, we adopted agile methodologies, breaking it into manageable segments to encourage continuous collaboration and improvement. We initiated the process by analysing business requirements and creating an initial information model. Subsequently, we designed UML database models and implemented them using SQL Workbench. After this, we crafted a basic state machine diagram, followed by the development of GUI components using Java Swing.

Our systematic approach prioritised a logical progression, seamlessly transitioning from the information model to the database model and then the state machine model for the user interface. This ensured a cohesive development process where each stage built upon the previous one. Our system is expected to perform the requested functions and demonstrate robustness against common cyber-attacks which we have discussed much more extensively later in this report.

To enhance collaboration, a GitHub repository was created at the start of the project for real-time updates. Initially, weekly two-hour meetings scheduled every Wednesday facilitated goal discussions and addressed any challenges. Pair-programming techniques were employed during database creation and UML designing. As the deadline approached, the frequency of meetings increased, ensuring close collaboration among team members and facilitating effective communication and problem-solving for timely project completion.

Interpretations and Assumptions

During our comprehension of the project brief, we had to make some inferences and assumptions. Here are some of the assumptions that we made:

- When a User signs up to the system, by default, they should have a customer role
- A user with the manager role has access to the same dashboard as a staff member but with an extra area of functionality limited to only managers
- Users wishing to use the system have an email address, and a physical address they can use to sign up with
- When confirming an order, the payment from the customer’s bank details will be taken from a third party source, i.e. a physical transaction or through a third party transaction system
- Staff members can delete products, if for example they no longer sell the product
- Staff and managers can filter and search in their respective dashboard areas, for better UX

- Products in the physical store can be found by just the product code and name if needed
- Staff and managers will be given guidance on how to use the system, so that not every UI element has to have over the top signposting that clutters the screen
- The only stakeholders involved are customers, staff, and managers
- The program will be deployed on machines with internet connection
- We assumed a hierarchical structure to the role system, so that for example rather than a staff member having the roles of customer and staff, they just have the role of staff which has an access level of one above a customer, meaning that they can access customer content and also staff content with a single role
- We assumed that box set quantities are not related to the quantities of the contents of the box set, as the box sets are already complete with contents before arriving at the store

UML Class Diagram of the Initial Information Model

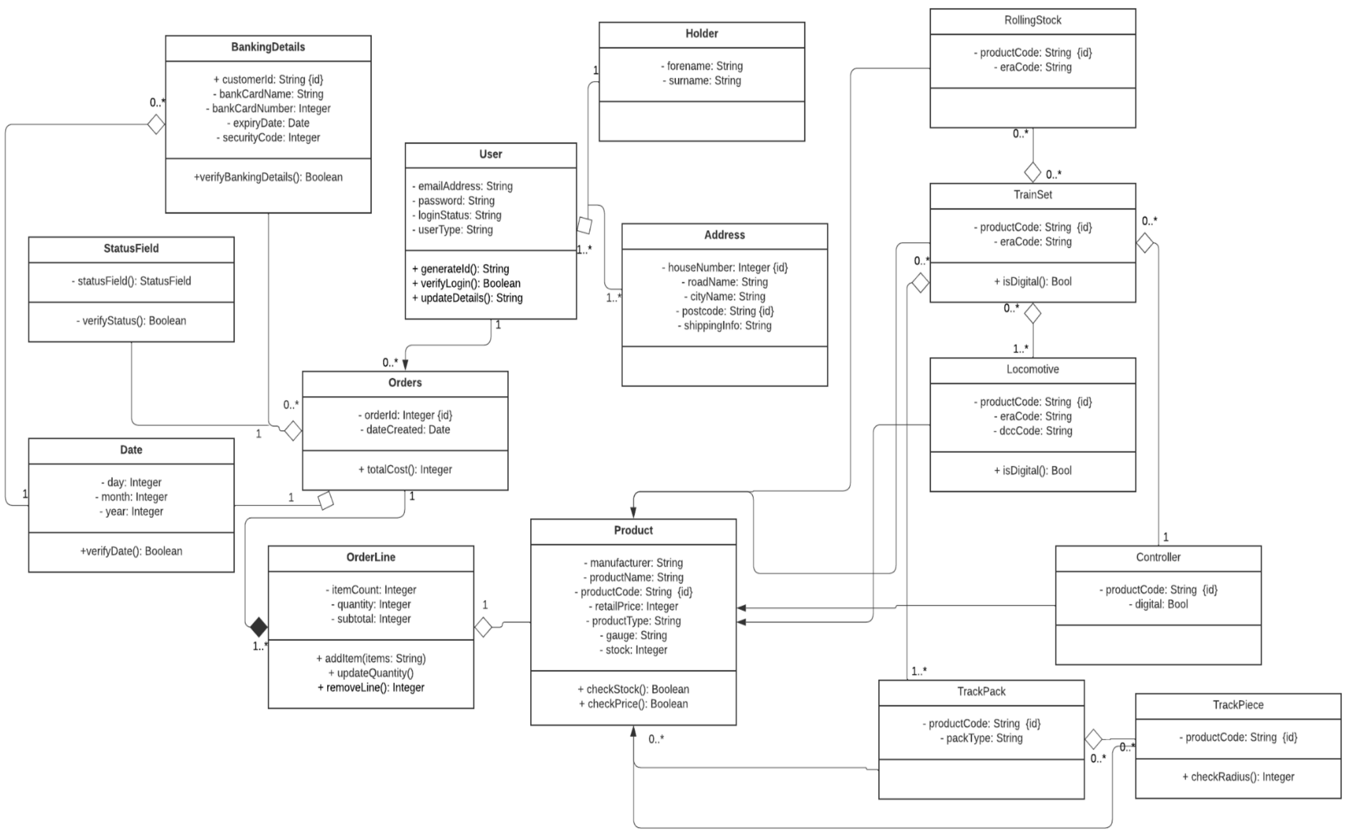


Figure 1 - Initial Information Model UML Diagram

The above image (figure 1) is a diagrammatic representation of our team's interpretation of the initial business model that was provided for the system. As a team, we took the provided business domain, business data, and business operation information of the system, and abstracted it down into multiple data objects that we represented as UML classes in a UML class diagram. During this process, we identified the type of attributes that each object required, as well as any methods that directly involved the manipulation or processing of the object's data. We then proceeded to identify the relationships between objects, documenting the type of relationship (i.e. composition, association, generalisation etc.), and any multiplicities that the relationship involved.

We deduced that the different types of products sold by the business could all be generalised to a Product superclass, but that each product would need its own subclass containing attributes and methods that only related to the specific product. The right side of the diagram shows these classes with their respective attributes and methods, and the different multiplicities between each type of product have been documented.

The left side of the diagram demonstrates the relationship between orders, products and users. In order to eliminate the many-to-many relationship between the product and order classes, it was clear to us that we would require a new class that provides a linking interface between the two classes. Therefore, we created the OrderLine class, which represents a single order line in an order, consisting of one product and its quantity.

Our final diagram presented here (figure 1) gave us an accurate and reliable representation of the initial system information model, and provided a solid foundation to begin our development process. All of our later models and program classes utilised this representation here to ensure that all aspects of the initial system were covered.

UML Class Diagram of the Normalised UML Database Model

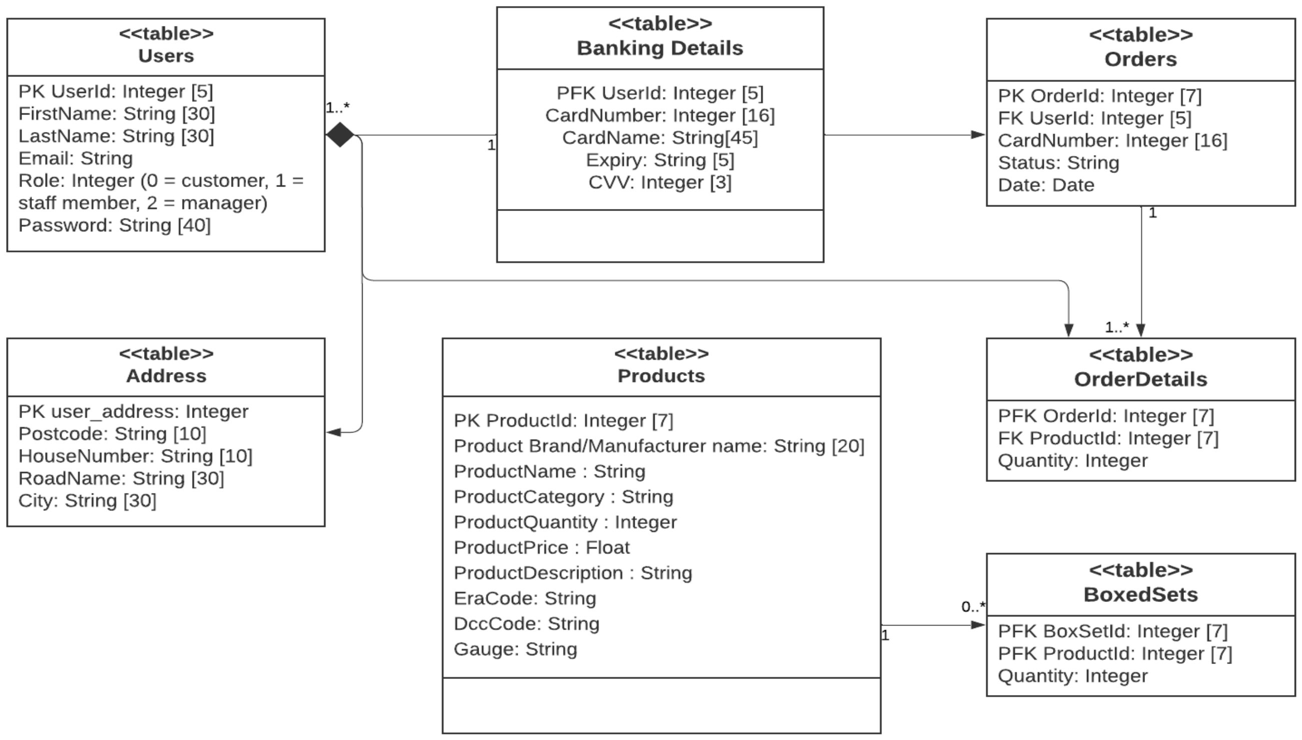


Figure 2 - Normalised UML Database Diagram

The database model intricately captures the core details essential for the seamless operation of the system. Upon user registration, all the information is stored in the User table, where the system-generated 'user_id' serves as the primary key, granting access to the associated data. The User table includes user-specific details such as user_id, first name, last name, email, and user role, denoted by integers (0 for customer, 1 for staff, and 2 for manager). The User table is connected to the Address table, wherein the user_address acts as the primary key, mirroring the user_ID. This table comprehensively archives address information, including postcode, house number, road name, and city.

The Banking table, linked to the User table through the user_ID as a primary-foreign key, serves as the repository for users' banking details. This table captures critical information like card name, number, expiry date, and CVV.

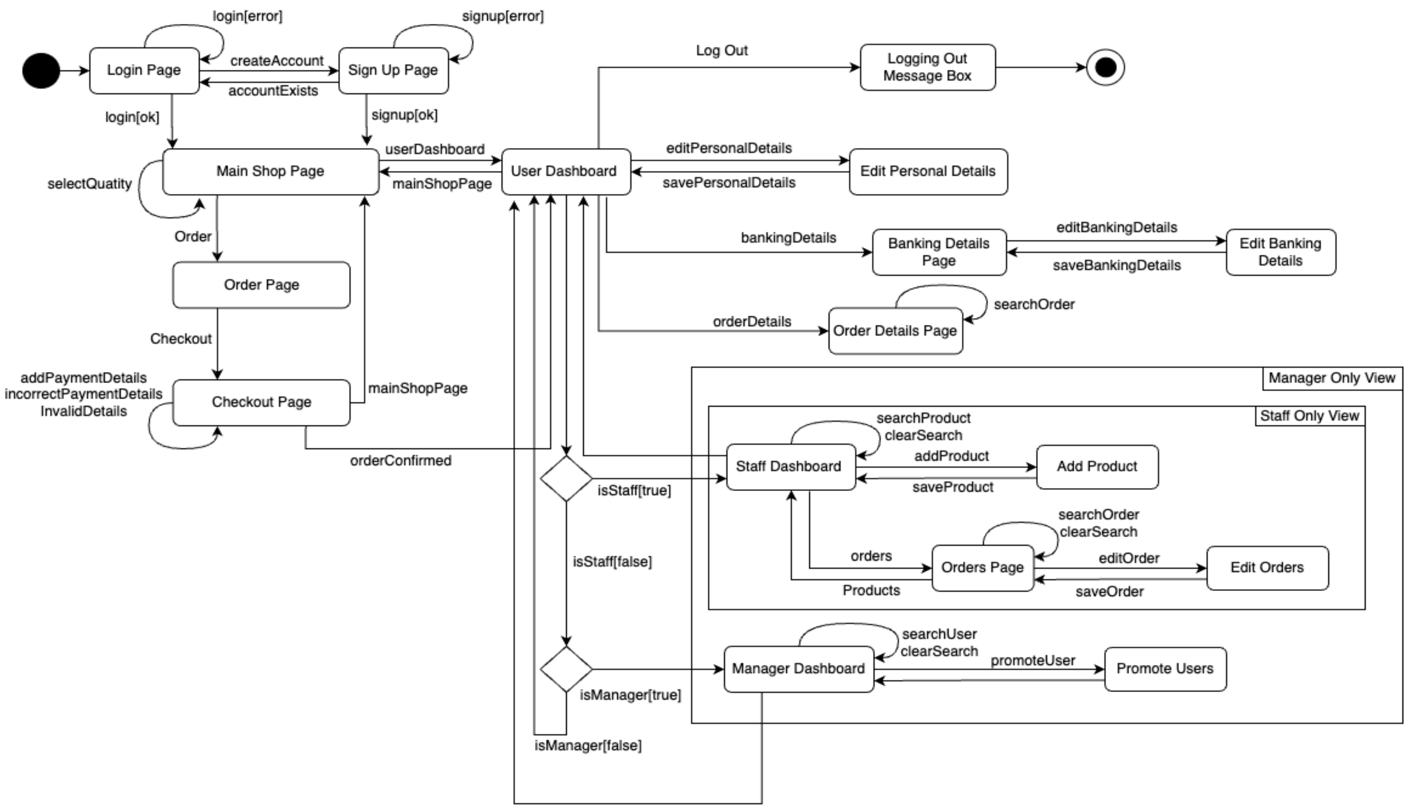
Order information is stored in the Orders table, linked to the User table via the user_ID as a foreign key. The primary key of this table is the system-generated order-ID, with additional details including card number, order status, and date.

The Products table holds all information regarding products available at Trains of Sheffield. Each product is uniquely identified by a product_ID, serving as the primary key. This table contains details regarding brand, name, category, quantity, price, description, era code, dcc code, and gauge. For boxed sets, a dedicated table is linked to the Products table, utilising the product ID as the primary foreign key. This additional table includes the box_set_ID and the number of boxed sets.

The Order Details table facilitates the retrieval of all information of an order. Linked to the Orders table through the order ID as the primary foreign key and the product ID as the secondary foreign key, this table captures intricate details, including the quantity of each product. The well-defined relationships and structure within these tables ensure a robust and organised foundation for the system's database model.

UML State Machine Diagram

Figure 3 - UML State Machine Diagram



The system mandates that all users, whether a customer or staff member, undergo a self-registration process. This involves providing personal details such as first name, last name, email, and password, along with their address details including house number, road name, city name, and postcode. Subsequently, a system-generated User ID is linked to the registered information. Upon successful account creation, users can log in using their email ID and password. The login page incorporates an intuitive mechanism, guiding users with non-registered email IDs to create an account through the signup page. In the event of entering invalid or incorrect information on either the login or signup page, an error message promptly notifies the user, ensuring a streamlined user experience.

Upon entering correct login/signup details, users are directed to the main shop page, which displays available products and their details. Here, users can add multiple products of varying quantities to their basket. The order button initiates a transition to the Order Page, providing users with options to add, edit, or delete items within their order. This page details the order comprehensively, including each product, quantity, individual prices, and the total sum. Upon satisfaction, users proceed to the checkout page to input payment details and confirm the order, after which further edits to the orders are restricted. Empty, invalid or incorrect payment details prompt the user and do not confirm the order..

Once the order is completed, users are redirected to the user dashboard, with an option to revisit the main shop page from the checkout. The user dashboard allows users to view and edit personal and bank details, along with a record of previous orders. For staff members, an additional staff dashboard is present, which offers extended functionalities to view, edit, and add products, as well as manage orders. If a user is a registered manager, they can access the manager dashboard, which contains all functionalities of staff members, plus the ability to view user details and execute promotions or dismissals between staff and users. Users can navigate back to the main shop page, or log out, from the user dashboard, thus terminating the system.

Screenshots

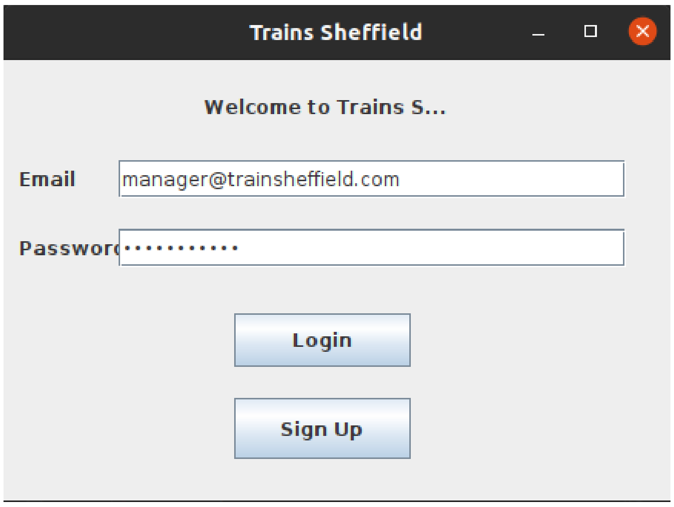
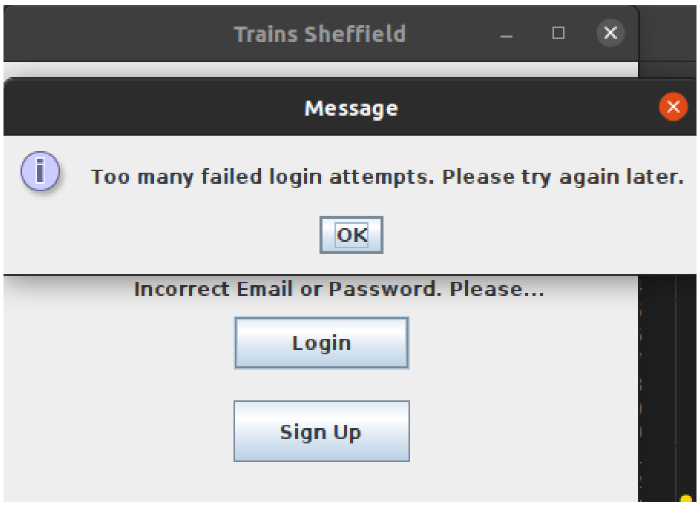


Figure 4 - Login Page

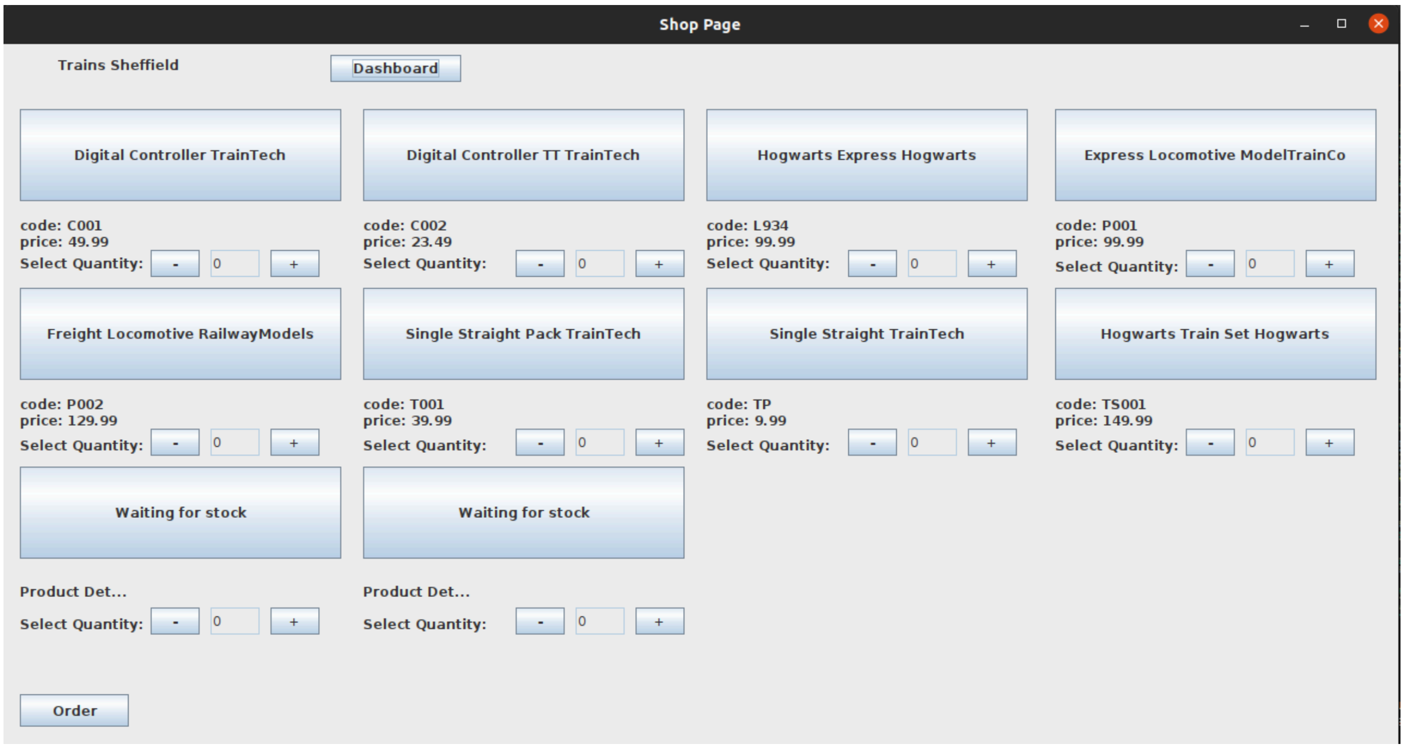


Too many failed login attempts. Please try again later.

Incorrect Email or Password. Please...

Figure 5 - Error Message after 5 failed login attempts

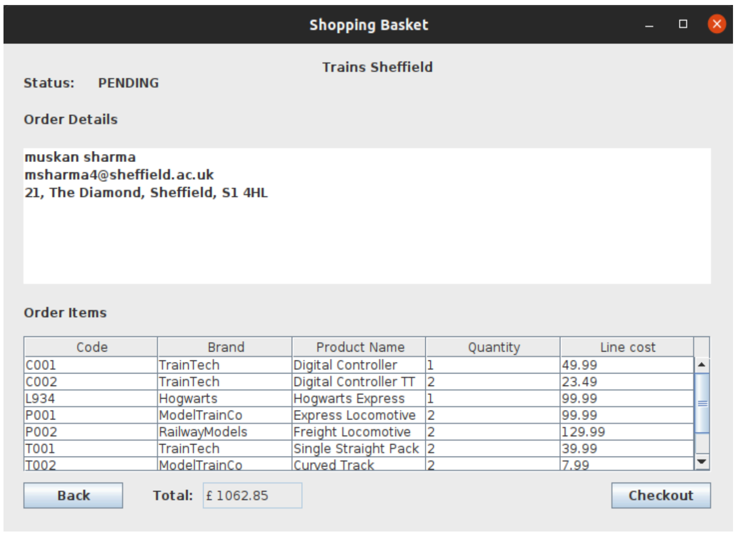
Our login page enables registered users to securely access accounts via email and password. Prioritising user information security, the system enforces a five-failed-attempt lockout mechanism for enhanced protection.



Digital Controller TrainTech	Digital Controller TT TrainTech	Hogwarts Express Hogwarts	Express Locomotive ModelTrainCo
code: C001 price: 49.99 Select Quantity: - 0 +	code: C002 price: 23.49 Select Quantity: - 0 +	code: L934 price: 99.99 Select Quantity: - 0 +	code: P001 price: 99.99 Select Quantity: - 0 +
Freight Locomotive RailwayModels	Single Straight Pack TrainTech	Single Straight TrainTech	Hogwarts Train Set Hogwarts
code: P002 price: 129.99 Select Quantity: - 0 +	code: T001 price: 39.99 Select Quantity: - 0 +	code: TP price: 9.99 Select Quantity: - 0 +	code: TS001 price: 149.99 Select Quantity: - 0 +
Waiting for stock	Waiting for stock		
Product Det... Select Quantity: - 0 +	Product Det... Select Quantity: - 0 +		

Figure 6 - Main Shop Page

Trains of Sheffield's main shopping page features a comprehensive display of available products, each accompanied by detailed information—name, code, and price. Users can seamlessly manage their selections, adjusting quantities with designated buttons. Upon completing their shopping experience, users can proceed to the order page by clicking the designated order button.



Trains Sheffield

Status: PENDING

Order Details

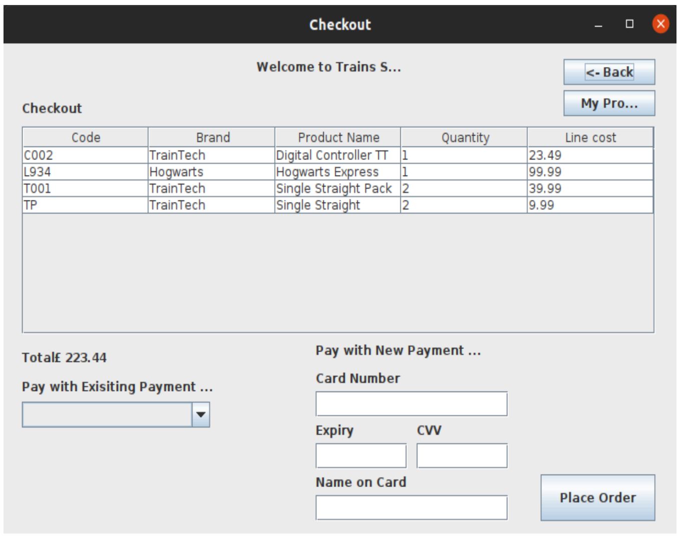
muskan sharma
msharma4@sheffield.ac.uk
21, The Diamond, Sheffield, S1 4HL

Order Items

Code	Brand	Product Name	Quantity	Line cost
C001	TrainTech	Digital Controller	1	49.99
C002	TrainTech	Digital Controller TT	2	23.49
L934	Hogwarts	Hogwarts Express	1	99.99
P001	ModelTrainCo	Express Locomotive	2	99.99
P002	RailwayModels	Freight Locomotive	2	129.99
T001	TrainTech	Single Straight Pack	2	39.99
T002	ModelTrainCo	Curved Track	2	7.99

Total: £ 1062.85

Figure 7 - Shopping Basket



Welcome to Trains S...

Checkout

Code	Brand	Product Name	Quantity	Line cost
C002	TrainTech	Digital Controller TT	1	23.49
L934	Hogwarts	Hogwarts Express	1	99.99
T001	TrainTech	Single Straight Pack	2	39.99
TP	TrainTech	Single Straight	2	9.99

Total£ 223.44

Pay with Exisiting Payment ...

Pay with New Payment ...

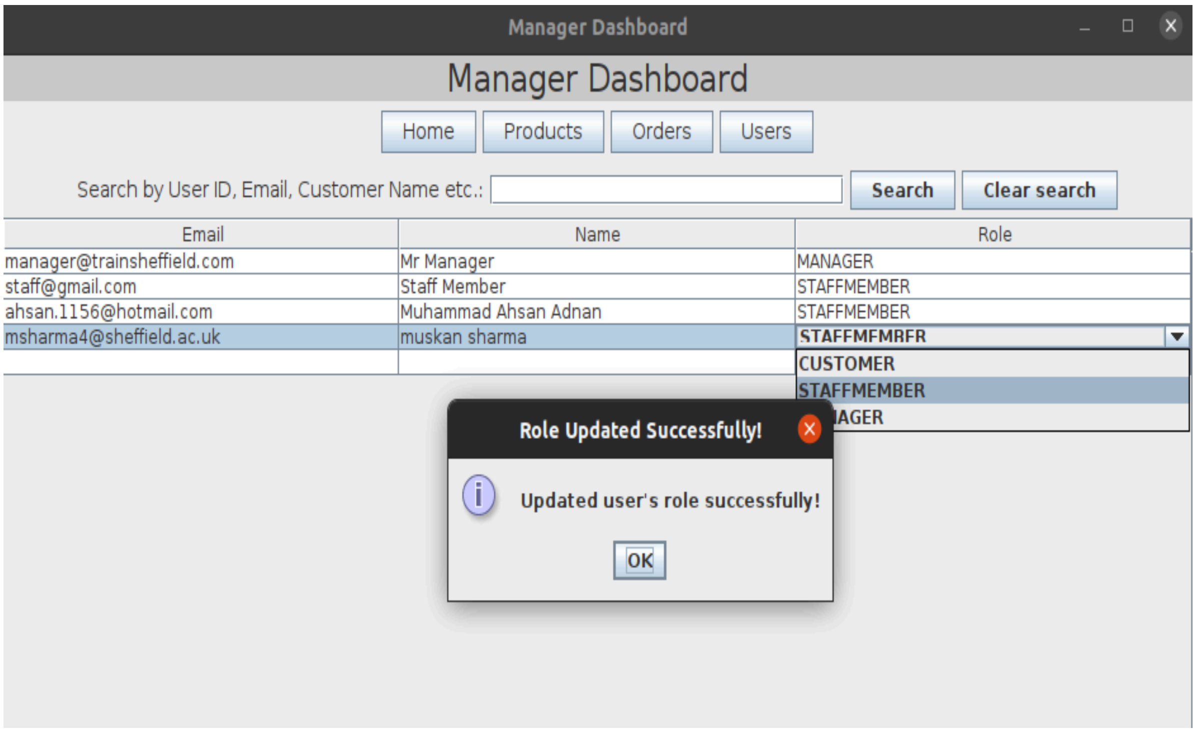
Card Number

Expiry CVV

Name on Card

Figure 8 - Checkout

The shopping basket page serves as a pivotal stage in our user journey, presenting comprehensive user and order details. Users enjoy a seamless experience, with the ability to effortlessly add, edit, or delete items, accompanied by a clear display of the total order cost. Transitioning to the checkout page, users can choose between registered payment methods or securely input new details, culminating in a streamlined and efficient order confirmation process.



Manager Dashboard

Search by User ID, Email, Customer Name etc.:

Email	Name	Role
manager@trainsheffield.com	Mr Manager	MANAGER
staff@gmail.com	Staff Member	STAFFMEMBER
ahsan.1156@hotmail.com	Muhammad Ahsan Adnan	STAFFMEMBER
msharma4@sheffield.ac.uk	muskan sharma	STAFFMEMBER

CUSTOMER
STAFFMEMBER
IAGER

Role Updated Successfully!

Updated user's role successfully!

Figure 9 - Manager Dashboard showing updating a user role

The Manager dashboard, exclusive to assigned personnel, provides role management capabilities. The diagram illustrates the Manager's ability to seamlessly update user roles, transitioning between customer and staff roles as needed.

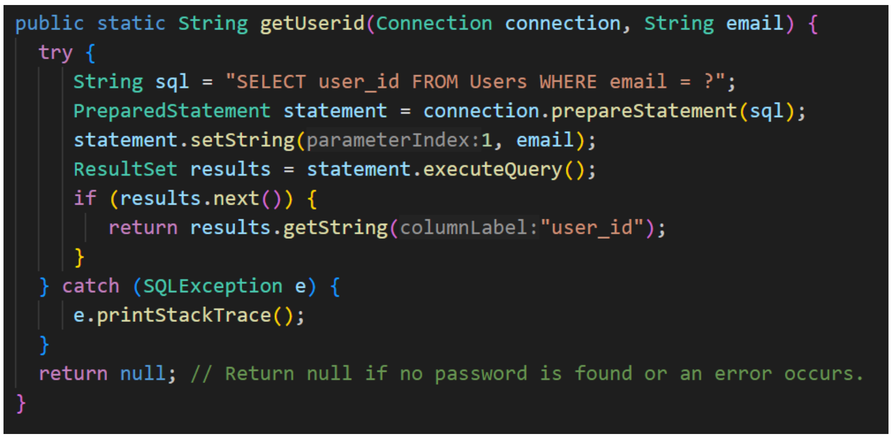
Security Considerations

We have implemented a MD5 hashing system for all the system passwords saved in the database. The reason for this is that hashing can only be done one way and can not be reverted. This means that even if a hacker gets access to the database they will only be able to get the hashed password which cannot be reverted back to its original form. So ultimately they cannot login to the system.

Secondly we have also implemented anti brute force attack measures. If a user tries to attempt to login more than 5 times with an incorrect password it will throw an error message and exit the program.

The users banking details have also been encrypted but we have used base64 encoding for this. Unlike for passwords we could not use a hashing system for this as the banking details need to be retrieved from the database and displayed to the user in case they want to pay with an existing payment method, which is why we needed an encryption method that could be reverted back to its original form.

To prevent our system from SQL Injection attacks we have made the use of PreparedStatements when interacting with the database, as shown below:



```java
public static String getUserid(Connection connection, String email) {
  try {
     String sql = "SELECT user_id FROM Users WHERE email = ?";
     PreparedStatement statement = connection.prepareStatement(sql);
     statement.setString(parameterIndex:1, email);
     ResultSet results = statement.executeQuery();
     if (results.next()) {
        return results.getString(columnLabel:"user_id");
     }
  } catch (SQLException e) {
     e.printStackTrace();
  }
  return null; // Return null if no password is found or an error occurs.
}
```

Figure 10 - Users.java (Line 41-54)

By using parameterised queries we can make sure that all inputs taken are treated as data and not executable code, which significantly reduces the risk of SQL injection attacks.

Lastly we have made sure that the system does not allow for any sort of privilege escalation. We have used a unique integer to identify each user role. Every user that logins is automatically assigned a customer role. The manager which has the highest role was manually assigned and created as a user in the database using SQL query. Secondly a staff member is assigned its role by the manager so they are also customers by default when they sign up. This also ensures that there is no chance for the system to glitch when a user signups where it accidentally assigns a customer a staff or manager role. We also covered SQL injection and how we have prevented that so a hacker cannot change the role value in the database either.