Engin Kılınç - 2698108

Esra Şekerci - 2698215

Designing a Second-Hand E-commerce Application

1 Project Case

The objective of our project is to design a second-hand e-commerce application, inspired by platforms like dolap.com, that facilitates the buying and selling of pre-owned items. The platform allows users to list items for sale, search for items to purchase, complete transactions, and leave reviews. This application aims to create a sustainable marketplace where users can find value in second-hand goods, thereby promoting recycling and reducing waste.

2 Project Scope

The project encompasses several key objectives. Firstly, user management features will allow users to register, log in, and manage their profiles with ease. Secondly, the platform will enable sellers to list products by providing detailed descriptions and images, ensuring that buyers have a clear understanding of the items available for purchase. The transaction handling process will be secure and efficient, covering aspects such as payment and shipping management. Additionally, a review system will be implemented, allowing buyers to review products they have purchased, thereby enhancing the overall quality and reliability of the marketplace. Lastly, a seller verification mechanism will be established to ensure the trustworthiness of sellers based on their transaction history.

The application is built using MySQL for database management, with backend functionalities implemented using MySQL stored procedures, triggers, and views. This technological stack will ensure robust data handling and efficient processing of complex business logic.

3 Capabilities

3.1 Database Schema Design

Our database schema is designed to support the comprehensive functionality of the e-commerce platform. The schema includes the following tables:

1. Users: This table stores user information, including registration dates and login details, ensuring secure and personalized access to the platform.
2. Products: Information about listed products, such as descriptions, images, and pricing, is stored here, enabling detailed and accurate product listings.
3. Transactions: This table records all transactions between buyers and sellers, facilitating the tracking and management of sales and purchases.

4. Reviews: User reviews for products are stored in this table, providing feedback and enhancing the credibility of listings.
5. SellerInformation: This table contains information about sellers, including their ratings and verification status, helping to establish trust within the marketplace.
6. Brands: Information about various product brands is stored here, allowing for brand-specific searches and categorization.
7. Messages: This table stores communications between users, supporting efficient and organized messaging within the platform.

The accompanying ER diagram and relation mapping provide a visual representation of the database schema, illustrating the relationships and interactions between different entities within the system.

3.2 Entity Relationship Diagram

The Entity-Relationship (ER) diagram visually represents the database structure of our second-hand e-commerce application. It includes entities such as Users, Products, Transactions, Reviews, SellerInformation, Brands, and Messages. The diagram illustrates how these entities interact with each other through relationships.

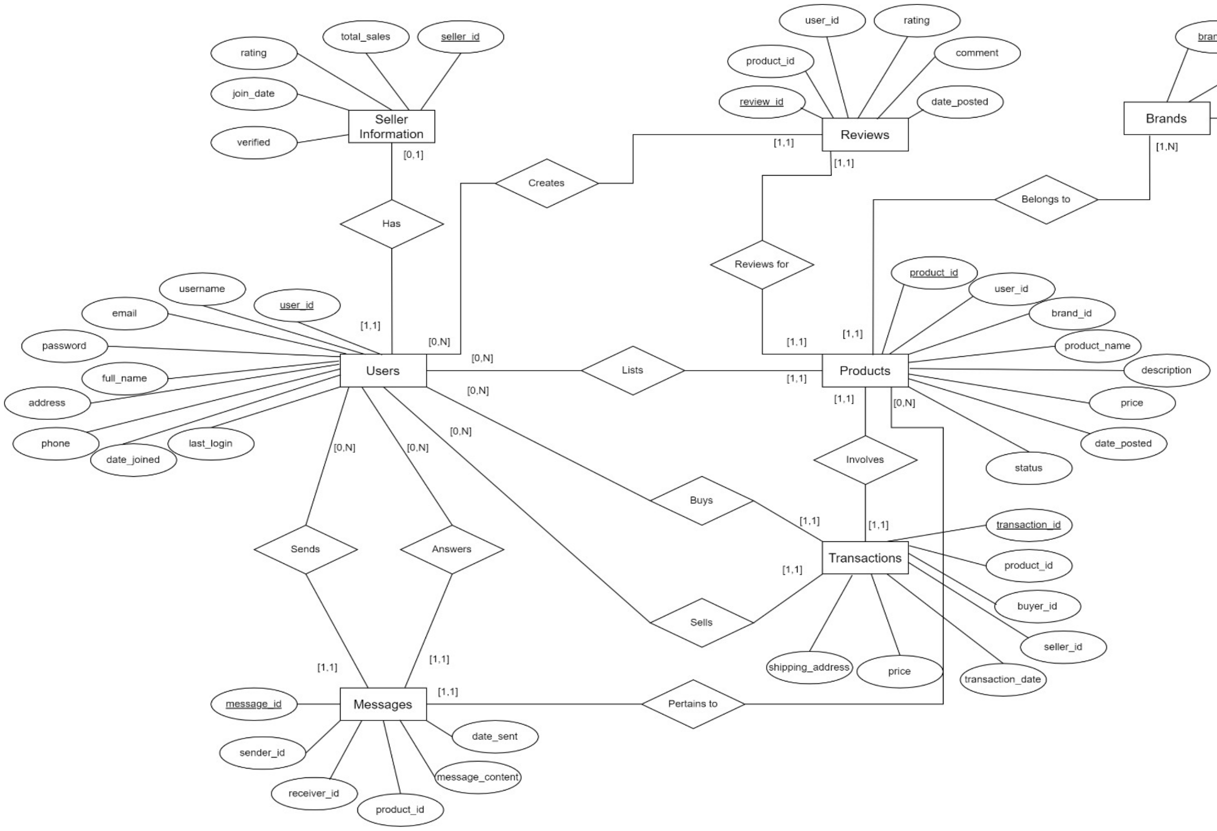
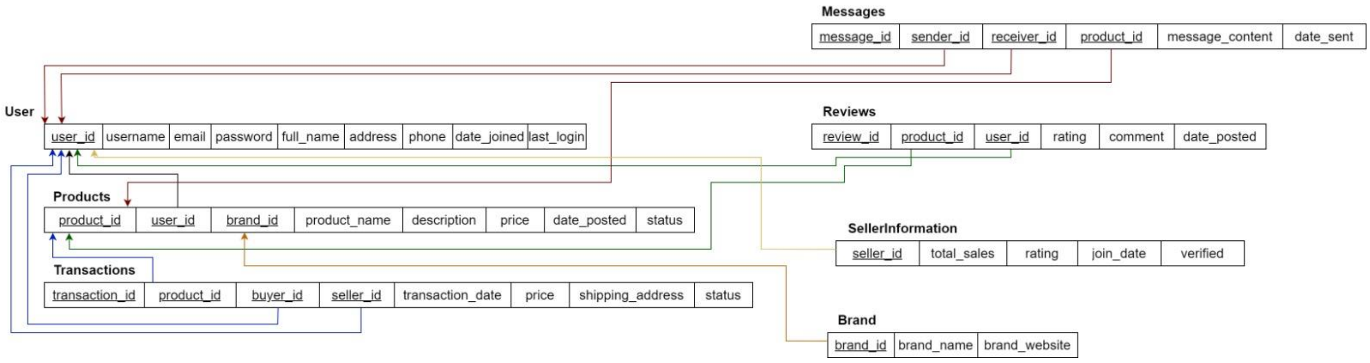


Figure 1- Entity Relationship Diagram

3.3 Relational Mapping

The relational mapping diagram provides a detailed view of how the database tables are interconnected. It includes primary keys, foreign keys, and the interactions between tables.



Messages
message_id | sender_id | receiver_id | product_id | message_content | date_sent
User
user_id | username | email | password | full_name | address | phone | date_joined | last_login
Reviews
review_id | product_id | user_id | rating | comment | date_posted
Products
product_id | user_id | brand_id | product_name | description | price | date_posted | status
SellerInformation
seller_id | total_sales | rating | join_date | verified
Transactions
transaction_id | product_id | buyer_id | seller_id | transaction_date | price | shipping_address | status
Brand
brand_id | brand_name | brand_website

Figure 2 - Relational Mapping

3.4 Complex Queries

Our application supports several complex queries to enhance functionality and provide valuable insights:

- Verified Sellers with High Ratings and Active Listings: Identifies verified sellers with a rating above 2.5 who have at least one active listing.
- Products Sold by Top-Selling Sellers with Specific Brands: Retrieves details of products sold by top-selling sellers for specific brands.
- Active Buyers with Reviews: Identifies active buyers who have provided a significant number of high-rated reviews.
- Sellers with No Completed Transactions: Identifies sellers who have no completed transactions but have listed products matching a certain keyword.

3.5 Analytics-focused View

We created a view DashboardAnalytics that aggregates various metrics for the dashboard analytics of the application. This view provides insights such as the number of new users, total products, pending and completed transactions, average product rating, total reviews, and information about the top user. This view is essential for monitoring the overall performance and user engagement within the platform.

3.6 Triggers

Triggers are essential for automating database operations and ensuring data integrity in our second-hand e-commerce application. We have created six triggers to handle various scenarios:

- Before Transaction Insert to Update Seller Information: This trigger executes automatically before inserting a new row into the Transactions table. It updates the SellerInformation table if the transaction status is 'completed', incrementing the total sales count for the seller.
- After Transaction Insert to Update Product Status: This trigger executes automatically after inserting a new row into the Transactions table. It updates the status of a product in the Products table based on the transaction status ('completed' or 'pending').
- After Review Insert to Update Seller Rating: This trigger executes automatically after inserting a new row into the Reviews table. It calculates the average rating of a seller based on reviews associated with completed transactions and updates the SellerInformation table accordingly.
- Enforce Review on Completed Transaction: This trigger enforces rules for adding reviews. It checks if a completed transaction exists for the product and buyer before allowing a review insertion. It also prevents multiple reviews for the same product by the same buyer.
- Update Last Login: This trigger updates the last_login field in the Users table to the current timestamp whenever the table is updated and the last_login field is not NULL.
- Verify Seller After Sale: This trigger executes after inserting a new row into the Transactions table. It checks if a transaction is marked as 'completed' and updates the seller's verification status in the SellerInformation table based on the number of completed transactions.

4 Limitations

While our application has many strengths, it also has some limitations. Scalability may be a concern, as the initial design might need optimization for handling a large number of users and transactions. Additionally, the current scope does not include integration with third-party payment gateways, which could limit the payment options available to users.

5 Assumptions

Our project is based on several assumptions. We assume that users have access to a stable internet connection and are familiar with basic e-commerce operations. Additionally, we assume that the server infrastructure can handle the expected load of users and transactions. Lastly, we assume that data privacy and security measures are in place to protect user information.

6 Optimizations and Improvements

To enhance the performance and reliability of our application, we have implemented several optimizations and improvements. Indexing has been added to frequently queried columns to improve performance. Foreign key relationships have been defined with cascading actions to maintain referential integrity. The schema adheres to normalization principles to reduce redundancy and improve data integrity. Finally, stored procedures have been used for complex business logic to maintain consistency and encapsulation.

7 Future Work

There are several areas for future work to enhance our application further. Scalability can be improved by implementing caching mechanisms to handle increased load and optimize query performance. Security features can be enhanced, including encryption for sensitive data and advanced authentication mechanisms. Additionally, a user-friendly front-end application can be developed to interact with the backend functionalities seamlessly.

8 Conclusion

This project provided valuable insights into designing and implementing a comprehensive e-commerce application. The challenges encountered and solutions implemented have strengthened our understanding of database design, query optimization, and the importance of maintaining data integrity in a real-world application. Our collaborative efforts resulted in a robust and scalable solution, ready to be extended and deployed in a production environment.