REQUIREMENTS

1.Notify me - Iustin

The system shall integrate with all other marketplace application components to ensure seamless communication and data consistency.

All notifications shall appear on a dedicated notification page, accessible to each user. All the notifications will have only a Title, a short description, a timestamp and a badge, more details will be revealed on another page, when the notification is clicked. Unread notifications shall be visually distinguished with a badge. Additionally, users shall receive pop-up alerts for new notifications while logged into the app. There will be a field indicating the number of unread notifications.

General info about the notification: Each notification shall include dynamic variables to personalize the content. These variables shall be replaced with actual values before the notification is sent. The recipient’s name, the category of the notification, the duration of the contract (if applicable), the name of the product referenced (if applicable), the unique id of the product (if applicable), the contract in pdf form (if applicable), the unique identifier of a product (if applicable), the bid value (if applicable), shipping details (if applicable) ,contract renewal state (if applicable) and the star rating received by a seller(if applicable) shall be dynamically included in notifications.

The system shall send various predefined notifications, including payment confirmation notifications for successful transactions, order shipping progress notifications to track shipping updates, and contract-related notifications such as renewal requests, renewal approvals or denials, and renewal options available for buyers and sellers. Additionally, the system shall notify users when a product becomes available, when a product is removed from the marketplace, when they are outbid on an item and the raiding received . These notification templates will be stored in a dedicated database table with relevant fields.

All notifications sent by the system shall be stored in the database for logging, tracking, and troubleshooting. The system shall store notifications in a table with specific schema fields, including a unique identifier for each notification, recipient id, the category of the notification, the full notification content, a timestamp indicating when the notification was sent, and an optional attachment field for cases where a file, such as a contract PDF, needs to be included.

To ensure reliability, the system shall log all sent notifications for verification. Before sending a notification, the system shall verify that all required variables are properly replaced, and if any required value is missing, the notification shall not be sent. The system shall be capable of handling up to 500 concurrent notifications send requests without performance degradation. Each notification shall be processed and sent within five seconds of being triggered. Additionally, notification database queries shall execute in under 200 milliseconds for efficient retrieval of past emails.

2.Payment - Sonia

When the Finalize Purchase selection is made, the system shall redirect the user to a Billing Information Page, where they must provide their details before completing the payment. The required fields include Full Name, Email Address, Phone Number, Address, and Postal Code, and may select a Payment Method. Users may also enter optional additional delivery information or special requests. The information fields shall be labeled clearly, with validation for required fields. These details must be validated before proceeding to the payment step. If any input fails validation, the system shall highlight the field and the issue (Name cannot be left empty, Phone Number must have 10 digits and start with “07”, Address cannot be left empty, Postal Code must have 6 digits and Email follows a valid format, e.g “example@domain.com”).

On the Billing Information Page, the system shall also display a summary of the order, listing each product from the basket along with its name, quantity, price, and any applicable taxes. The total cost shall be clearly displayed, including the product subtotal, any applicable warranty tax, and the delivery fee.

The system shall apply a warranty tax of 20% on borrowed products, which will be included in the final order total. After the finalization of the order, the tax amount shall be stored in the wallet, but will remain inaccessible to the user. Once the contract period ends, the user will gain access to these funds in their wallet.

Additionally, for orders below 200 lei, a fixed delivery fee of 13.99 lei shall apply, otherwise the delivery fee is 0 lei.

If the basket contains borrowed items,the system shall introduce an additional contract setup step in the Billing Information Page. Under each borrowed item, an option to select the start and end date(in DD/MM/YYYY format) of the borrowing period shall appear. The system shall validate that the end date is after the start date and within the maximum borrowing period allowed for that product. Once confirmed, a contract is automatically generated containing key details such as price, seller and buyer information, borrowing period, product details, and terms and conditions.

The system shall support two payment methods: Cash on Delivery and Card Payment. If the user selects Cash on Delivery, no card details will be required, and the order will be processed immediately. If the user selects Card Payment, they must enter their Email, Cardholder’s Name, Card Number, Card Expiry Date (MM/YY), CVC Code. Payment validation will ensure that the card number is correct (using the Luhn algorithm, a checksum formula used to validate credit card numbers), the expiration date is in the future, the CVC code is correctly formatted and the email follows a valid structure.

The system also supports a Wallet feature for bidding. Refilling the wallet is considered a card-only order, following the same billing and payment process. Users can select a top-up amount, using the interface from the CtrlAltElite(User) team and will be sent to the Billing Information Page. Wallet balance will be updated upon successful payment. At the end of a bidding process, the winning user shall purchase the product only using the funds from their wallet.

Upon successful payment, the system shall redirect the user to a Payment Complete page, displaying their Order ID, Total Amount Paid, Payment Method Used, and Order Status ("Processing"). A "Continue Shopping" button shall also be available. The order shall be automatically added to the user’s order history, including details about purchased and borrowed items, contract information (if applicable), and payment details. Additionally, a Payment Confirmation Notification will be sent to both the buyer and seller. This notification shall include order details, confirmation of payment, and, if applicable, attached contracts for borrowed products. If the payment fails, the system shall display an error message (a page) specifying the issue (e.g., "Invalid Card Number" or "Insufficient Funds") and allow the user to retry.

For performance, the Billing Information Page must load within 2 seconds on a broadband connection (minimum 25 Mbps). The payment gateway must return a success or failure response within 5 seconds for 95% of transactions. If a payment gateway failure occurs, the system must allow users to retry without losing their entered data. Failed payments shall not be deducted from the user's account.

The payment system shall support up to 500 concurrent users processing payments during peak hours without system degradation. The infrastructure shall be scalable to accommodate increased transaction volumes.

3.Contract Handling - Darius

-The system shall automatically generate a contract when a borrowing agreement is made between a seller and a buyer. The generated contract shall include essential details such as the price, seller and buyer information, borrowing period (with start and end dates in ISO 8601 format), product details (including product name, ID, and category), and terms and conditions, which will default to a template unless otherwise specified, in which case the nullable field custom terms will be completed. The contract will be created using the pdfQuest .net library. The system shall validate that all required fields, including price, seller and buyer information, borrowing period, and product details, are populated. Additionally, it shall ensure that the start date is earlier than the end date. If validation fails, a specific error message will be displayed, clearly identifying the problematic field and describing the nature of the issue, for example, “Error 101: Missing Borrowing Period.”

-Predefined Contract content will be held in a table of its own containing an ID, Predefined_Contract_Type and Predefined_Contract_Content.

-The content of the Contract will contain formatted variables for fields that will be completed with the afferent data from the Contract table.

-In cases where contract generation or PDF storage fails, the system shall inform the user via an error message and log the error details, including the error type, timestamp, and affected contract data, for debugging purposes. Contract details shall be stored in a dedicated database table with the following columns: contractID (unique identifier), price, sellerID and buyerID (references to user records), productID (reference to product records), startDate and endDate (borrowing period), contractStatus, and Contract Content with possible values of active, renewed, or expired, and a nullable warranty field in case of borrow and renewal, and the pdf file as contract. Typical database load conditions are defined as 50 concurrent users performing standard query operations such as retrieval, updates, and inserts.

-PDF files will be stored in their own separate table connected to the Contracts table. The relation will be 1:1, they will be stored in a separate table to ensure the fast operation of the Contracts table.

-Performance metrics dictate that the Users shall be able to retrieve stored contracts within 2 seconds under typical database load conditions. This functionality shall be integrated with the existing borrowing agreement module. If the borrowing agreement module fails during integration, the system will notify the user and log the failure for review. The retrieval will be executed via a download button on the notification widget which will in turn download the pdf file for contract.

4.Order Tracking - Ilinca

Order Tracking: The system shall allow users to track their orders from the moment payment is confirmed until successful delivery. The system shall display the current status of the order, utilizing major checkpoints: "Processing", "Shipped", “In Warehouse”, "In Transit", "Out for Delivery", and "Delivered" along with timestamps and location(the location field is nullable, it can be empty if “Shipped”, “In Transit” or “Out for Delivery”). The order status page shall load and display tracking information of the specified order based on the unique OrderID

Order Notification: The system shall send a notification to the seller and the buyer once an order has been placed, both containing the order details and delivery address as well as a brief notification that the order is “Processing”

Simulated Order Checkpoints: The system shall automatically generate estimated checkpoints based on predefined time intervals. These checkpoints shall provide users with an approximation of their order’s progress. The system shall ensure order status updates occur automatically at predefined time intervals unless manually changed by the seller or admins.

Manual Order Checkpoints: The system shall allow sellers or administrators to manually update the order details (order status and the details of the checkpoints: description, location, timestamp) through a dedicated interface with a field for editing each attribute. The system shall also allow sellers or admins to revert the order to the previous checkpoint or to advance it to the next checkpoint.

User Notifications: The system shall send notifications to the buyer when the order progresses through key checkpoints: “Shipped”, “Out for Delivery”. Notifications shall be sent to the buyer within 30 seconds of the status update.

Estimated Delivery Date: The system shall display an estimated delivery date based on predefined time rules: “Processing” - immediately after order confirmation, “Shipped” - 24h after “Processing”, “In Transit” - 1 hour after “Shipped”, “In Warehouse” - 24h after “In Transit”, “Out for Delivery” - next day at 8am after “In Warehouse”, “Delivered” - 12h after “Out for Delivery”. So the initial estimated delivery date should be 3 days after the order is placed.

Error Handling: If an order status update fails, the system shall log the error, notify the seller or an admin through a notification that contains the OrderID where the error occurred and the error details for manual correction.

Verification: The system shall ensure that order tracking updates match the expected format before displaying them to users: status should be part of the statuses domain ("Processing", "Shipped", “In Warehouse”, "In Transit", "Out for Delivery", and "Delivered") and should represent the current status of the order, checkpoints list contains valid checkpoints (the timestamp is in DD/MM/YYYY hh:mm format), estimated delivery date is a valid DD/MM/YYYY date after the order placement date

5.Renew Contract - Bianca

The system shall allow users to renew contracts for borrowed products before the expiration date, ensuring a seamless extension of the borrowing period. Renewals shall be subject to predefined conditions, including availability (provided by the Products team, MarketMinds), maximum borrowing limits set by the seller and seller approval.

The system shall allow contract renewal only if the product is still available for borrowing and the seller agrees to the extension. A renewal request can be made up to 7 days before the current contract expiration date (but not in the last 2 days before the contract expires). If the renewal request is approved, the system shall update the old contract with the new borrowing period and notify the buyer, seller and the waitlist about the successful renewal. If the renewal request is denied, the system shall notify the user and provide the reason (e.g., “Product not available” or “Seller declined renewal”). If a renewal request is not made between 2 and 7 days before the contract expires, the product shall automatically become available to waitlisted users.

The new borrowing period must be within the product’s maximum allowed duration. The renewal cannot exceed the total borrowing period allowed for that product.If the seller sets custom renewal conditions, the system shall enforce them (e.g., only one renewal allowed per contract). The contract ID, buyer ID and product ID must match an existing active contract before renewal is allowed.

The system shall update the status of the renewed contract to reflect the change (e.g., Active -> Renewed). Before applying the renewal, the system shall store the previous contract details within the same Contracts table, maintaining a reference to the original contract. The updated contract shall be stored as a new PDF document, preserving all legal details.

The system shall trigger a system notification when:

1. A renewal request is submitted (notification to buyer & seller).

2. A renewal is approved (buyer & seller receive updated contract details in notifications & waitlist).
3. A renewal is denied (buyer receives a notification with the reason for rejection).
4. The contract is about to expire without renewal (buyer is informed via notification and the product is released to the waitlist).

Notifications shall include the variables {username, product name, previous end date, new end date, renewal status} and shall be stored in the Notifications Table with the following attributes: notification ID, sender, receiver, notification type, content, timestamp and attachment (nullable).

The Renew Contract page shall load within 2 seconds under standard conditions (25 Mbps broadband). If the contract renewal process fails, the system shall log the error with a timestamp, the affected contract ID and the error type, notify the user about the failure and allow the user to retry the renewal request if the error is recoverable.

6.Order History - Laza Bogdan

-Contract access, for borrow + basket for new and used products (order ID)

-automatically adding the contract when creating + date

-every user has an order history

- filter history

The system shall provide registered users with access to their order history, which will include all past transactions involving products in their account, whether borrowed or purchased. The order history shall display the following details for each order: product photo(jpg), product name(string), product price(integer greater than 0), order date(dd/mm/yyyy), and a button to "See Order Details" for each past order that, when clicked, will display the corresponding borrowing contract(pdf) for borrowed products or order details, such as distributor(string), delivery address(street, postal code, apartment number) and review stars(from 1 to 5 which will be an average mean between the past users' reviews(each from 1 to 5 stars)) all displayed in a new window, for new products.

The system shall automatically add an entry for both the borrowed and purchased products along with their order details(contract or (distributor, delivery address and review stars))

The order history will also include a search bar from where the user will be able to search the desired order by product name and a drop out menu with predefined choices for sorting after the date(all orders/orders from last 3 months/orders from last 6 months/orders from 2025/orders from 2024)

If no orders are found matching the user's criteria or if there are no orders in the period mentioned by the user, the system shall display a message: "No orders found"

The order informations will be kept in a database in an order table with columns such as OrderID, ProductID, OrderType(new/used/borrowed), Distributor, DeliveryAddress, Rating(between 1 and 5), ContractDetails(only for borrowed products) that will support crud operations such as insert delete and update.

The order history shall be accessible via the user’s profile page, with all historical data loaded within 3 seconds on a broadband connection (minimum 25 Mbps).

7. Waitlist Handling - Labo Nikole

The system shall manage a waitlist for borrowed products that are currently unavailable. Users who wish to borrow an unavailable product shall have the option to join a waitlist and also a user who is already in a waiting list will be able to leave it at any time. The system shall track the order of users in the queue and ensure timely notifications when the product becomes available.

When a product becomes available for the waitlist(2 days before contract expires), the system shall automatically notify the first user in the queue. If the user does not claim the product within 24h , the system shall notify the next user in line until the product is successfully assigned. If none of these 2 users are willing to borrow the product the whole waitlist will be notified with 12 hours before the current contract expires and each of them can claim it (first to confirm will take it).

If a contract renewal occurs for a product on the waitlist, all users in the queue shall receive a notification. The notification shall provide an option to remain in the waitlist or leave the queue. Users who choose to leave the waitlist shall be automatically removed from the queue.

If a product is permanently removed from the marketplace, the system shall notify all users on the waitlist. All users shall be automatically removed from the waitlist to ensure data consistency and avoid unnecessary queue entries.

Database Storage

The waitlist shall be stored in a dedicated database table with the following schema:
- Waitlist ID: Unique identifier for each waitlist entry, formed by combining User ID and Product ID.
- User ID: Reference to the user who joined the waitlist.
- Product ID: Reference to the unavailable product.
- Position in Queue: Indicates the user's order in the waitlist.
- Joined Timestamp: Records the time when the user joined the waitlist.
- Available Again: Stores the estimated date when the last active contract for this product is set to expire.

Notifications

The system shall send predefined notifications based on waitlist events. The following notification types shall be triggered:
- Product Available: Sent to the first user in the queue with 48h before the product becomes available for borrowing, if the first user doesn't confirm, then to the second user in the queue with 24h before. If none of these will confirm their next borrow the notification is sent to the entire waitlist with 12 before the last contract expires.
- Contract Renewal: Sent to all waitlisted users when the product's borrowing contract is renewed.
- Product Removed: Sent to all waitlisted users if the product is permanently removed from the marketplace.