Comprehensive Exercise Report

Team ETIK of Section 101

Kayra Akel 221ADB169, Irmak Kurekci 221ADB194, Tuna Yalcin 221ADB180, Elif Cokcan 231ADB021, Efehan Aras 221ADB079

Requirements/Analysis

Week 2

Journal

The following prompts are meant to aid your thought process as you complete the requirements/analysis portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- After reading the client's brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.
    - **Description:** A second-hand marketplace platform for Riga, designed to help students (starting from age 16) and businesses learn to grow their economy, make money for themselves, and easily buy and sell pre-owned fashion and accessories through a user-friendly mobile website.
    - **Known Requirements:**

1. User Accounts & Authentication – Secure registration, login, and profile management.
2. Listing System – Users can upload product images, descriptions, and set prices.
3. Search & Filters – Advanced search with category, brand, size, and price range filters.
4. Secure Payment System – Integrated online payment and escrow service to ensure safe transactions.
5. Chat & Notifications – In-app messaging and notifications for buyer-seller communication.
6. Order Management – Status tracking for buyers and sellers.
7. Delivery & Pickup Options – Shipping integration or local pickup coordination.
8. Admin Panel – Moderation tools, analytics, and customer support management.
9. Review & Rating System – Users can rate sellers and provide feedback.
10. Multi-Language Support – Support for at least Latvian and English.
11. Product Controller – A controller who will verify the safety, condition, and whether the product is new or used before listing it for sale.

- After reading the client's brief (possibly incomplete description), what questions do you have for the client? Are there any pieces that are unclear? After you have a list of questions, raise your hand and ask the client (your instructor) the questions; make sure to document his/her answers.

1.
Interviewer: How do you currently buy or sell second-hand items (like clothes, accessories, etc.) in Riga?
Interviewee: I usually use platforms like Facebook Marketplace and local groups. Sometimes I also check out some international sites like eBay, but it can be tricky with shipping fees and language barriers.

2.
Interviewer: What challenges do you face when using existing platforms for second-hand shopping or selling?
Interviewee: The biggest issue is the lack of trust. It's hard to know if the item is in good condition or if the seller is reliable. Plus, there's a lot of spam and irrelevant posts that make browsing frustrating. In

addition, there’s no such a website for selling&buying second hand in Riga. Also I couldn’t help but mention, I study away from my parents since I was 15-16 years old. For me, such an website could saved my life to earn some money on my own.

3.
Interviewer: How important do you think it is for students, especially from the age of 16, to have a platform that helps them make money and learn about growing their own economy?
Interviewee: It’s really important! A lot of students don’t have access to part-time jobs, and an easy-to-use platform for selling things would not only help them earn some extra money but also teach them valuable business skills. Like I mentioned before, I could do something on my own with this website. Beside that, it will really help me to understand and give me to encourage to improve the business skills.

4.
Interviewer: Would you find it helpful to have a specific platform for selling and buying and also focused on second-hand items where products are reviewed for safety and condition before being listed?
Interviewee: Yes, definitely! Knowing that items have been checked for safety and condition would make me feel much more confident in buying second-hand goods. It would eliminate a lot of the stress that comes with uncertainty

5.
Interviewer: How do you feel about the idea of a product controller verifying items before they are available for sale? Do you think it would help build trust with buyers?
Interviewee: I think it’s a great idea. It would definitely help build trust and ensure the platform maintains a good reputation. Sometimes it’s hard to trust that items are as described or photographed orginially, so having someone check them would make the buying experience much smoother.

6.
Interviewer: What features would you expect or find useful in a second-hand marketplace app designed for students and young entrepreneurs?
Interviewee: I’d like to see features like easy payment processing, secure chat options for negotiating, and a good review system for buyers and sellers. Also, some educational content about managing finances and selling skills would be awesome.

7.
Interviewer: Would you be interested in selling products through an online platform that offers the opportunity to build your own small business? If so, what kind of products would you want to sell?
Interviewee: Yes, I’d be interested. I know that 16 years old me also interested in this website but I am still interested in such a kind of act. I’d probably sell clothes, books, or tech gadgets. It would be cool to have the chance to grow a small business and maybe even offer unique or homemade items to other students.

8.
Interviewer: How much do you think a platform like this could help you manage your finances or teach you about entrepreneurship?
Interviewee: It could really help me get better at budgeting and pricing products. We normally had a entrepreneurship class back in RTU 2nd year. But having a way to track sales and expenses would definitely teach me about entrepreneurship and managing money in a practical way.

9.
Interviewer: What do you think about the multi-language support (Latvian and English) in such a platform? Would that be an important feature for you?
Interviewee: Yes, that would be really important. Since many students speak both Latvian and English, it would be great to have the platform accessible in both languages. It would also attract international students who might want to use it. Just an addition I could say that it should also contain Russian.

10.
Interviewer: What concerns or improvements would you suggest for a platform that aims to support students in making money and learning to manage their economy?
Interviewee: I'd like to see low fees or no fees for students, especially for new users. The platform should be intuitive and not complicated. It might also be nice to have educational resources for entrepreneurship, like tips for pricing products or how to create a good product description. But I am sure that you can do that.

- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
    - Some libraries needs to be studied (i.e. UI libraries)
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
    - - Main users of this software will be the people who want to sell any second hand materials or handmade products from home. It can be anybody starting from the age of 16 regardless the education level or gender. .
- Describe how each user would interact with the software
    - Seller: Sellers will be able to create their shop pages and manage them like a social media account. Sellers can advertise their products and contact buyers, they can also contact and work with other shops.
    - Buyers: Buyers will be able to search and find the advertisements of products, contact with sellers and buy stuff online. They will also be able to comment on products.
- What features must the software have? What should the users be able to do?
    - Sign-up/Login with e-mail
    - Profile and product listing management
    - Search
    - Verified payment and transaction system
    - Admin dashboard
    - Chat and notifications
- Other notes:
    - <<Insert notes>>

Software Requirements

Project Title: SKAPIS

Project Description:

SKAPIS is an online marketplace platform developed specifically for Riga, its main purpose is to empower students and small businesses to buy and sell pre-owned or handmade products. It aims to support financial independence and entrepreneurship through a safe, user-friendly website. It will provide item verification, secure payments, and a learning-friendly environment to its users. With SKAPIS, users can list items, communicate with sellers/buyers, transact money securely, and receive educational prompts on managing finances or running a small business. SKAPIS also supports multiple languages, making it accessible for local and international users alike.

2. Functional Requirements

1-User Authentication and Profile Management: The platform will allow users to securely register and log in. After authentication, users will have access to personal dashboards to manage and edit their profiles.

2-Product Listing System: Sellers can upload products with multiple images, write descriptions, choose appropriate categories, and set pricing. Each listing will be visible to all users once verified.

3-In-App Messaging: Buyers and sellers can communicate directly through a secure in-app chat feature, enabling negotiation, product inquiries, and coordination without leaving the platform.

4-Search and Filtering: Advanced search capabilities will help users find items efficiently using filters such as product category, brand, size, price range, and location.

5-Payment Integration and Escrow Service: The system will include a secure payment gateway (e.g., Stripe or PayPal). An escrow mechanism will ensure that payments are only released when both parties are satisfied.

6-Admin Panel and Moderation: An administrative dashboard will provide backend tools for monitoring user activities, verifying listings and users, responding to reports, and viewing analytical data.

7-Product Controller Verification: A dedicated product controller will check each listed item for condition, authenticity, and safety before it becomes visible in the marketplace, ensuring trustworthiness and quality.

8-Review and Rating System: Users will be able to rate each other after transactions and leave feedback on listings. This fosters accountability and transparency within the community.

9-Notifications: Users will receive real-time notifications for new messages, order updates, promotional alerts, and important system announcements.

3. Non-Functional Requirements

1-Security: The system must protect user data by encrypting all sensitive information, including passwords, to ensure safe access and compliance with privacy standards.

2-Usability: The interface must be simple, intuitive, and user-friendly, specifically designed for students and users with minimal technical experience.

3-Performance: The website should load its core features (e.g., home page, search, chat) in under three seconds to ensure a smooth and responsive user experience.

4-Availability: The platform must guarantee 99% uptime, minimizing system downtime and ensuring consistent accessibility for users.

5-Scalability: The system must be capable of scaling efficiently to accommodate increased usage, especially during periods of high demand such as seasonal promotions or school events.

4. User Stories

As a 16-year-old student, I want to be able to sell my used clothes on a trustworthy platform so I can earn my own money.
As a seller, I want the items I list to be reviewed and verified before publication to ensure trust and credibility.
As a buyer, I want to search for products by size and price so I can quickly find what I need.
As an admin, I want to manage and verify user and product data efficiently to keep the platform secure.
As a user, I want to chat securely within the app so I don’t have to rely on external tools or platforms.

5. Other Notes

The platform should minimize or eliminate transaction fees for students, especially during early use. It may include pop-up tips or short educational modules related to entrepreneurship, pricing strategy, and product presentation. In future versions, SKAPIS could consider integrating with university systems or student ID verification to strengthen student-oriented features.

Black-Box Testing

Instructions: Week 4

Journal

Remember: Black box tests should only be based on your requirements and should work independent of design.

The following prompts are meant to aid your thought process as you complete the black box testing portion of this exercise. Please review your list of requirements and respond to each of the prompts below. Feel free to add additional notes.

- What does input for the software look like (e.g., what type of data, how many pieces of data)?
  - User-entered text (e.g., email, password, product descriptions, chat messages)
  - Uploaded files (e.g., product images)
  - Selection values (e.g., product category, language selection)
  - Button clicks (e.g., “Add to cart”, “Send message”)
  - Filter criteria (e.g., price range, size, brand)
  - Payment credentials or tokenized info (via Stripe or PayPal)

- What does output for the software look like (e.g., what type of data, how many pieces of data)?
  - Success/failure messages (e.g., “Login Successful”, “Invalid password”)
  - Updated UI state (e.g., product page loads with filtered results)
  - Notifications (e.g., “Order confirmed”, “New chat message”)
  - Displayed data (e.g., listings, order tracking status)
  - Redirects to relevant pages (e.g., after checkout or login)

- What equivalence classes can the input be broken into?
  - Valid email vs invalid email
  - Password length > 8 vs < 8
  - Price range: valid (e.g. 1–999) vs invalid (e.g. -10, 100000)
  - Uploaded image: valid format/size vs unsupported/too large
  - Valid vs expired payment token
  - Filtered search with 0 vs >0 results
  - Reviewed vs not-reviewed product listing

- What boundary values exist for the input?
  - Password length: minimum 8, maximum 64 characters
  - Product description length: min 10 characters, max 1000 characters
  - Price: minimum 0.01, max 9999.99
  - Image file size: max 5MB
  - Chat message length: max 250 characters
  - Rating values: 1 to 5

- Are there other cases that must be tested to test all requirements?
  - Login with wrong password or nonexistent account
  - Posting a product with missing required fields
  - Trying to filter with no input (empty filters)
  - Chat message sent to a deactivated account
  - Changing language in the middle of a transaction
  - Submitting a payment without confirming the address

    - Attempting checkout with an empty cart
    - Product controller rejecting a faulty listing

- Other notes:
    - All tests should simulate real user behavior without accessing the source code. Each test must validate correct system output for given user-facing inputs.

Black-box Test Cases

Use your notes from above to complete the black-box test plan section of the formal documentation by writing black box test cases (other than actual results since no program currently exists). Remember to test each equivalence class, boundary value, and requirement.

Test ID	**Description**	**Expected Results**	**Actual Results**
Test-1	Login with valid credentials	Redirect to user dashboard	Redirects to index.html after storing token
Test-2	Login with wrong password	Show error message: "Invalid password"	Error message shown and login prevented
Test-3	Register with invalid e-mail format	Show error: “Invalid e-mail address”	Frontend validation prevents submission
Test-4	Register with short password	Show error: “Password must be at least 8 characters”	Form displays error, registration not allowed
Test-5	Upload product with all valid fields	product is listed successfully	Product appears in product list after creation
Test-6	Upload product with missing title	Show error: “Title is required”	Form displays error, title is required
Test-7	Upload image larger than 250MB	Show error: “File too large”	Form displays error, file too large

Design

Instructions: Week 6

Journal

Remember: You still will not be writing code at this point in the process.

The following prompts are meant to aid your thought process as you complete the design portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- List the nouns from your requirements/analysis documentation.
    - User
    - Seller
    - Buyer
    - Product
    - Shop
    - Order
    - Message
    - Notification
    - Review
    - Payment
    - Rating
    - Price
    - Account
    - Description
    - Condition
    - Image
    - Chat
    - Feedback
    - Profile
- Which nouns potentially may represent a class in your design?
    - User
    - Product
    - Shop
    - Order
    - Message
    - Notification
    - Review
    - Payment
    - Chat
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - <<Insert answer>>

Attribute	Class it Belongs to
Name	User, Product
E-mail	User
Password	User
Role(buyer/seller)	User
Profile info	Profile
Price	Product
Description	Product
Condition	Product
Image	Product
Rating	Review
Feedback	Review
Account ID	User or Payment
Status	Order
Timestamp	Message, Notification

- Now that you have a list of possible classes, consider different design options (***lists of classes and attributes***) along with the pros and cons of each. We often do not come up with the best design on our first attempt. Also consider whether any needed classes are missing. These two design options should not be GUI vs. non-GUI; instead you need to include the classes and attributes for each design. Reminder: Each design must include at least two classes that define object types.
    - Design Option 1
      Classes and Their attributes:
        1. User (Base Class)
            - id
            - name
            - e-mail
            - password
        2. Buyer (Inherits from user)
            - orderhistory
            - saveditems
        3. Seller (Inherits from User)
            - shopname
            - productslisted
        4. Product
            - id
            - title
            - description
            - price
            - condition
            - image
            - sellerid

5. Order
    - id
    - productid
    - buyerid
    - status
    - timeplaced
6. Review
    - id
    - reviewerid
    - sellerid
    - rating
    - feedback
7. Message
    - id
    - senderid
    - receiverid
    - content
    - timesent
8. Notification
    - id
    - userid
    - message
    - timecreated

a) Pros:
    - It would have clear separation between buyers and sellers
    - In this design each role can have specific actions and data
    - It would be easier to manage different responsibilities

b) Cons:
    - One user can't easily be both a buyer and a seller
    - Some repeated code for user-related features
    - More complicated design for real-life flexibility

- Design Option 2
  Classes and Their Attributes
  1. User
      - id
      - name
      - e-mail
      - password
      - isBuyer(true/false)
      - isSeller(true/false)
      - profileinfo
  2. Product
      - id
      - title
      - description
      - price
      - condition
      - image
      - sellerid
  3. Shop
      - id
      - ownerid
      - shopname

        - lisfofproducts
    4. Order
        - id
        - productid
        - buyerid
        - status
        - timeplaced
    5. Review
        - id
        - reviwerid
        - userbeingreviewedid
        - rating
        - feedback
    6. Message
        - id
        - senderid
        - receiverid
        - content
        - timesent
    7. Notification
        - id
        - iserid
        - message
        - timecreated
    8. Payment
        - id
        - orderid
        - amount
        - paymentstatus

a) Pros:
    - A user can be both buyer and seller easily
    - Fewer classes means easier to understand and manage
    - Good for real-life usage where users may change roles

b) Cons:
    - It would need extra checks. For example only sellers can add products
    - Some logic like what users can do, may get a little messy
    - More ‘if’ statements in the code to handle different roles

- Which design do you plan to use? Explain why you have chosen this design.

  We are planning to use Design Option 2 because it is more flexible, simple to build, scalable , easier to maintain and better for user experience.

  A single user can act as both a buyer and a seller using the same account, which matches real-life usage ,especially for students who might want to buy some items and sell others at the same time. List the verbs from your requirements/analysis documentation.It’s easier to manage one user class with extra role flags (like isSeller or isBuyer) than to deal with separate buyer/seller classes and inheritance. Also, as the platform grows, we can easily add more features (like admin roles or delivery partners) without needing a complex class structure. Fewer classes and less repeated code make the system easier to understand, update, and debug.Users won’t need separate accounts to switch roles, which makes the platform more user-friendly, especially for young users like students.

- List the verbs from your requirements/analysis documentation.
    - Register
    - Login
    - Upload

  - View
  - Edit
  - Delete
  - Create
  - Search
  - Filter
  - Chat
  - Send
  - Recieve
  - Buy
  - Sell
  - Rate
  - Review
  - Track
  - Notify
  - Pay
  - Comment
  - Update
  - Manage
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
  -

Method(Verb)	**Belongs to Class**	**What It Does**
register()	User	create a new user
login()	User	log into the system
updateProfile()	User	edit user profile details
createShop()	Shop	open a new shop as a seller
uploadProduct()	Product/Shop	add a product to the platform
editProduct()	Product	modify product details
deleteProduct()	Product	remove a product listing
search()	Product	find products using keywords
filterProducts()	Product	narrow down product listings
placeOrder()	Order	buy a product
cancelOrder()	Order	cancel an order in progress
trackOrder()	Order	check order delivery status
makePayment()	Payment	process a transaction
sendMessage()	Message	send a message to another user
recieveMessage()	Message	receive a message

startChat()	Chat	begin a new chat session
notifyUser()	Notification	send notification
createReview()	Review	submit a product or seller review
rateUser()	Review	assign rating
comment()	Review	add a comment to a product
manageShop()	Shop	control shop settings and listings

    - <<Insert answer>
- Other notes:
    - Role flexibility is important, users should be able to switch between buyer and seller roles without needing separate accounts.
    - Security is a must: User data, payments, and chats must be securely stored and transferred (e.g., via encryption and secure login methods).

Software Design

Class Structure Overview

The system follows an object-oriented structure consisting of several core classes that represent the business logic of the website. These classes include User, Store, Product, Cart, Order, and Negotiation. Each class contains relevant attributes (fields) and public methods to encapsulate the core functionalities of the system.

Class Descriptions

Class: User

Fields:

- id: int
- name: string
- surname: string
- email: string
- password: string
- phone: string
- birthday: string

Methods:

bool login(string email, string password);

void updateProfile(UserDetails details);

Class: Store

Fields:

- **id: int**
- **ownerId: int**
- **name: string**
- **description: string**

Methods:

void create();

Store getByOwner(int ownerId);

Class: Product

Fields:

- **id: int**
- **storeId: int**
- **name: string**
- **description: string**

- **price: double**
- **imageUrl: string**

Methods:

void create();

void update();

void delete();

List<Product> getAll();

List<Product> getByStore(int storeId);

Class: Cart

Fields:

- **userId: int**
- **items: List<CartItem>**

Methods:

void addItem(int productId, int quantity);

void removeItem(int productId);

void clear();

List<CartItem> getItems();

Class: Order

Fields:

- **id: int**
- **userId: int**
- **productList: List<Product>**
- **total: double**
- **createdAt: DateTime**

Methods:

void placeFromCart();

void placeSingle(int productId);

List<Order> getHistory(int userId);

Class: Negotiation

Fields:

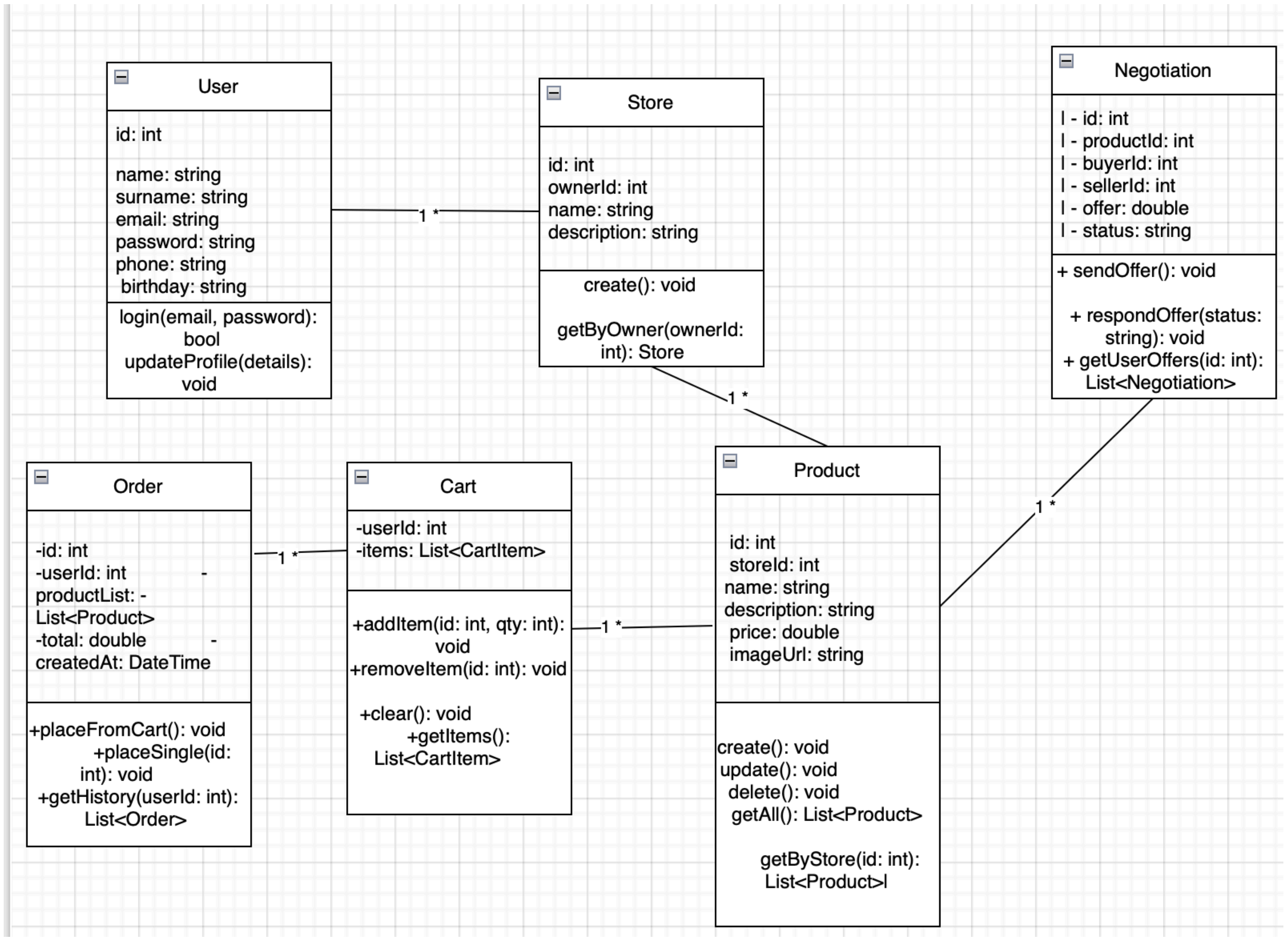
- **id: int**
- **productId: int**
- **buyerId: int**
- **sellerId: int**
- **offer: double**
- **status: string // e.g., "pending", "approved", "rejected"**

Methods:

void sendOffer();

void respondOffer(string status);

List<Negotiation> getUserOffers(int userId);



Implementation

Instructions: Week 8

Journal

The following prompts are meant to aid your thought process as you complete the implementation portion of this exercise. Please respond to each of the prompt below and feel free to add additional notes.

- What programming concepts from the course will you need to implement your design? Briefly explain how each will be used during implementation.
    - **Object-Oriented Programming (OOP)**
      Object-oriented programming is essential for organizing the website into modular, reusable components. Classes like User, Product, Cart, and Order will encapsulate data and related behaviors using attributes and methods. This structure ensures separation of concerns and clean code architecture.
    - **RESTful API Communication**
      The frontend will interact with the backend via RESTful API endpoints using fetch() or axios. Each operation (e.g., sign in, get products, add to cart) will map to a specific HTTP method (GET, POST, PUT, DELETE).
    - **Token-based Authentication (JWT)**
      JSON Web Tokens (JWT) will be used to securely authenticate users. Upon login, a token will be stored in localStorage and sent with each authenticated request in the Authorization header.
    - **Event-Driven Programming**
      Frontend interactions such as form submissions, button clicks, and dynamic content loading will use JavaScript event listeners (e.g., addEventListener('click', ...)).
    - **Asynchronous Programming (Promises / async/await)**
      API calls from the frontend will be made using async/await to handle asynchronous behavior. This ensures smooth user experiences and proper handling of loading states or errors.
    - **Conditional Rendering**
      JavaScript will be used to dynamically update the UI depending on the user state (e.g., logged in vs logged out), cart content, or backend responses.
    - **Data Structures (Arrays, Objects)**
      Arrays will be used to manage lists of products, cart items, and orders. Objects will represent individual entities like a user, product, or negotiation offer.
    - **Modular Code Organization**
      The project will be structured with separate HTML pages, JavaScript files (e.g., signin.js, cart.js, script.js), and backend API routes for maintainability.

Implementation Details

This website was implemented using a client-server architecture. The backend was developed using Node.js and Express, deployed on Render. The frontend was built with HTML, CSS, and Vanilla JavaScript, hosted locally or on a static file host.

The frontend and backend communicate through RESTful API endpoints, where user authentication and actions like managing products, cart, and orders are handled via HTTP requests.

1. Open the signin.html page in your browser.
2. If you don't have an account, click "Sign Up" to go to the registration form.
3. After signing in, you'll be redirected to the main page (index.html) where you can:
   - View all available products
   - Click on your profile icon to manage your account, addresses, or billing
   - Create your own shop via the "Create Shop" link

Shopping & Orders

- You can add products to your cart from any product list page.
- Open the cart to review and remove items if necessary.
- You can either:
  - Place a direct order on a single item, or
  - Checkout the entire cart using the "Place Order" button

For Sellers

- After creating a shop, you can add your own products.
- You can view, update, or delete your own products.
- You will also receive negotiation offers from buyers and can approve or reject them.

Authentication

- The system uses JWT token-based authentication.
- You must be logged in to access protected pages like:
  - Product creation
  - Cart actions
  - Orders
  - Negotiations
- If your token expires or is missing, the system will redirect you to the signin page.

Testing

Instructions: Week 10

Journal

The following prompts are meant to aid your thought process as you complete the testing portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- Have you changed any requirements since you completed the black box test plan? If so, list changes below and update your black-box test plan appropriately.
  - Yes, several requirements and implementations were adjusted during the testing process based on observed behavior and validation results. These changes led to updates in the test plan and helped strengthen system security and user experience.
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.

Class: User

- **Equivalence Classes:**
  - Valid email & password → Login successful
  - Invalid email/password → Error message
  - Duplicate email → Registration error
- **Boundary Values:**
  - Password length: min 8, max 64
  - Email string length: max 255
  - Empty fields
- **Paths to Test:**
  - Sign in
  - Sign up
  - Update profile (with partial or full info)

Class: Store

- **Equivalence Classes:**
  - Valid store creation → Success
  - Missing fields → Error
  - Unauthorized access → Rejection
- **Boundary Values:**
  - Store name: min 1, max 100 characters
  - Description: up to 1000 characters
- **Paths to Test:**
  - Create store (authenticated vs unauthenticated)
  - Retrieve user store

Class: Product

- **Equivalence Classes:**

  - Complete valid product info → Listing created
  - Missing fields or invalid price → Error
  - Unauthorized creation → Rejection
- **Boundary Values:**
  - Price: 0.01 – 9999.99
  - Title: max 100 characters
  - Description: 10 – 1000 characters
  - Image file size: max 5MB
- **Paths to Test:**
  - Add product
  - View all products
  - View by store
  - Update/delete product (authorized vs unauthorized)

Class: Cart

- **Equivalence Classes:**
  - Valid product added → Appears in cart
  - Invalid product or no auth → Error
  - Remove non-existent product → Silent fail
- **Boundary Values:**
  - Number of cart items: 0 – 50
  - Quantity per item: 1 – 99
- **Paths to Test:**
  - Add to cart
  - Remove from cart
  - View cart
  - Clear cart

Class: Order

- **Equivalence Classes:**
  - Valid checkout from cart → Success
  - Empty cart → Error
  - Unauthorized request → Blocked
- **Boundary Values:**
  - Total price: 0.01 – 9999.99
  - Number of products in order: 0 – 100
- **Paths to Test:**
  - Place order from cart
  - Place order for one product
  - View order history

Class: Negotiation

- **Equivalence Classes:**
  - Valid offer amount → Sent successfully
  - Offer to wrong product → Rejected
  - Accept/reject logic → Status updated

- **Boundary Values:**
  - Offer amount: 0.01 – very high value
  - Status: “pending”, “accepted”, “rejected”
- **Paths to Test:**
  - Submit negotiation offer
  - Respond to offer
  - View user or seller negotiations

Testing Details

All tests were performed strictly through user-facing interfaces (e.g., HTML forms, buttons, inputs) without accessing or relying on source code.
Inputs included:

- Text fields (e.g., email, passwords, product descriptions)
- File uploads (product images)
- User selections (filters, categories)
- Button interactions (submit, delete, send)

Outputs included:

- Success/failure messages
- UI changes
- Redirects
- Displayed listings or errors

Each test case validates visible system behavior based on the stated functional requirements, not implementation.

Presentation

Instructions:Week 12

Preparation

The following prompts are meant to aid your thought process as you complete the presentation portion of this exercise. It is recommended that you examine the previous sections of the journal and your reflections as you work on the presentation as it is likely that you have already answered some of the following prompts elsewhere. Please respond to each of the prompts below and feel free to add additional notes.

- Give a brief description of your final project
    - SKAPIS is an online second-hand and homemade product marketplace platform tailored for Riga. It aims to empower people and small businesses by providing them with a secure and user-friendly space to sell and buy products, enhancing financial independence and encouraging entrepreneurship.
- Describe your requirement assumptions/additions.
    - Users are either buyers or sellers, each with distinct access and permissions.
    - Authentication and token-based login are assumed necessary for security.
    - Sellers can manage their shops, while buyers can browse, purchase, and negotiate.
    - Photos are required for product listings, with a 5MB file size limit.
    - All UI elements are assumed to work responsively across devices.
- Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
    - Agile was chosen because it allows continuous feedback from the client and faster iteration.
    - This supported our evolving design—especially after issues arose in integration and testing stages.
    - Frontend and backend designs were modularized to allow easier updates and fixes.
- How did the extension affect your design?
    - Updating the database schema
    - Adding secure chat functionalities
    - Extending the backend with negotiation logic
    - Adjusting the frontend UI to support interactive offers
      This significantly influenced both data handling and UI design.

- Describe your tests (e.g., what you tested, equivalence classes).
  Black-box testing focused on user behavior and form validation.

    - Classes tested included:

        - User: login/signup validation, duplicate emails
        - Shop/Product: valid entries, missing fields, unauthorized edits
        - Cart: add/remove limits, boundary quantities
        - Order: valid checkout, empty cart rejection
        - Negotiation: valid offers, offer updates

    - Boundary values and equivalence classes were defined for each input type.
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
    - Importance of token-based security and input validation
    - Time management under pressure and adapting roles flexibly in a team
    - Handling backend failures and debugging deployment issues
    - Learned how to implement a full client-server application with real-world features

    - Improved understanding of the Agile development process
- What functionalities are you going to demo?
    - User authentication: Sign In / Sign Up
    - Seller features: Create shop, edit shop & billing info, add product
    - Buyer features: Buy product, apply filters, negotiate with seller

- Who is going to speak about each portion of your presentation? (Recall: Each group will have ten minutes to present their work; minimum length of group presentation is seven minutes. Each student must present for at least two minutes of the presentation.)
    - **Tuna Yalcin**: Opening, Motivation, Methodology, Requirements
    - **Kayra Akel**: Technologies,Implementation,Design and SRS
    - **Efehan Aras**: Testing, Problems Faced/Lessons
    - **Elif Çokcan**:Demo walkthrough and conclusion/future work
    - **Irmak Kürekci**: Demonstration Video and prototype

- Other notes:
    - Ensure screen recording or live demo is ready and working
    - Emphasize real use cases from a student or small business perspective
    - Prepare backup slides in case demo environment fails