Revised Spec - Modoya

I use a pink background to highlight the sections that I revise.

▼ Nov. 2 Update

- **Application Logic (V1.0 Completed)**
  The system calculates both separate monthly_rent and buyout_price for every item. The shopping cart allows users to dynamically configure their order, including updating the rental duration and switching the item's action between RENT and BUY.
  - **Shopping Cart Management**: Persistent cart functionality is implemented using Flask Sessions, supporting dynamic item additions, configuration updates, and removal.
  - **Checkout Simulation**: The `checkout` endpoint successfully handles order finalization, calculates the total cost based on the current cart configuration, simulates the generation of a unique `order_id`, and clears the session.
- **Next Focus**
  1. User Filtering: Implement the functional Style/Color/Category filtering mechanism on the product browsing page.
  2. User Accounts: Begin building the user account system for managing rental history and favorites.
  3. Public Marketplace: Implement the C2C listing feature (create_c2c_listing).

🟧 General Description

Modoya (used to be “LoopLiving”) is a furniture platform designed for young, style-conscious people on a budget, as well as anyone who cares about

sustainability. We've observed that many college students purchase cheap, fast furniture at the beginning of the school year, only to discard it on the roadside at the end of the semester, even when it's still usable. This creates a significant amount of waste. Modoya aims to solve this problem with a core mission: "*to make a stylish lifestyle more accessible while promoting environmental sustainability.*"

The platform has two main parts:

- Official Designer Rentals
  This is a service run by the platform, offering designer and stylish furniture. Users can choose to either rent furniture by the month or buy it outright.
- Public Second-hand Marketplace
  This is a space for users to easily buy and sell their own used furniture, which helps give items a second life and reduces waste.

🛠️ How it Will be Built (The Tech Side)

👉🏻 The initial focus will be on developing a **website**. The core backend logic will be tested using a command-line interface. A web app will be considered only after the core features are mature.

Backend	Python Flask	
Frontend	HTML, CSS, JavaScript	
Dataset	A simulated furniture database (stored in JSON format)	Note: Cannot find any readily available free furniture APIs/dataset for now.
Data Handling	Pandas	
Prototyping	Google Colab for testing Figma for UI designing	
Future Nice-to-Have: AI Suggestion Agent	1. Morden Style: Focuses on modern interior design principles like space optimization, aesthetics, and functionality.	2. Feng Shui Advisor: Blends traditional principles with AI to offer actionable advice, helping to harmonize the user's living space.
External APIs	Gemini 2.5 Pro or GPT-5 Pro	

🟨 Task Vignettes / User Activity Flow

👩🏻‍🦽 Story: Sarah's Rental Experience

> Sarah opens the Modoya website and sees a list of popular, curated furniture. On the left side of the page, she uses the filters. She picks "Mid-Century Modern" from the "Style" dropdown menu and adjusts the "Price Range" slider to a maximum of $200/month. The page updates instantly with furniture that matches her choices (being handled by JavaScript).
>
> She clicks on a picture of an “Eames Lounge Chair”, which takes her to the product detail page. She sees two prices (monthly rental and a price to buy it now). She chooses a "six-month" rental term and clicks "Add to Cart." She then clicks the cart icon in the top-right corner.
>
> On the checkout page, she selects "Home Delivery," chooses a delivery time for next Wednesday afternoon from a small calendar, and confirms her address. She enters her credit card info and clicks "Complete Rental." A "Success!" message appears, and the new order details are added to her personal account page.

🛠️ Technical Details:

- Requires a structured furniture database with filterable fields such as "style," "brand," "type," and "rent/sale price.”
- The user account system must be able to manage current rentals, rental history, lease expiration dates, and wishlist/favorites.
- The backend needs to handle rental order generation and status tracking (e.g., pending delivery, in-use, pending return).
- Logistics will be simulated; the system will only need to update the status.

[NEW Table]

Sarah's Action	Status	UI Implementation
Sees list of furniture	Completed	`index.html` displays all products with images, monthly rent, and buyout prices.

Sees two prices (Monthly Rental and Buyout)	Completed	Item cards display `Monthly Rent` and `Buyout Price`.
Clicks "Add to Cart"	Completed	The `add_to_cart` route adds the item to the session.
Clicks the cart icon	Completed	`view_cart` route loads `cart.html`
In Cart: Edits duration, switches RENT/BUY	Completed	`cart.html` features forms to update duration and switch the item's action (`set_rent` / `set_buy`).
Clicks "Complete Rental"	Completed	The `checkout` route processes the order.
"Success!" message appears	Completed	Redirects to `checkout_complete.html` showing the Order ID and Total Charged.

🟩 Technical Flow

Data Flow: User → Flask Web → Database

- **User Interaction (Frontend):** Users interact with the Flask application through their web browser to browse, filter, and place orders.
- **Application Logic (Backend):**
  - **Receive Requests:** Flask receives HTTP requests from the user (e.g., "show all mid-century modern chairs").
  - **Database Query:** The application queries the SQLite/JSON database to retrieve the corresponding furniture data.
  - **Business Logic:** It processes the logic for rentals/purchases, calculates prices, and updates the user's dashboard.
  - **Render Page:** It inserts the queried data into an HTML template and returns the final web page to the user.
- **Data Storage:**
  - **Furniture Database:** Stores all detailed information about the furniture.
  - **User Database:** Stores user account information and their rental/purchase history.

Data Structures

- `furniture_dataframe`: A Pandas DataFrame that holds all the furniture information.
  - [Completed] Loaded and processed by `module.py`'s `get_all_items()`, forming the primary furniture data list.
- `user_filters_dict`: A dictionary to keep track of the user's current filter settings (e.g., style, price).
- `shopping_cart_list`: A list of items the user has added to their cart.
  - [Completed] Implemented using Flask's `session['cart']`, storing item ID, rental duration (`duration`), and action type (`order_type`).
- `favorites_list`: A list of item IDs that the user has saved to their wishlist.
- `reviews_dict`: A dictionary to store user reviews for items or sellers.

Core Functions

- `load_furniture_data()`: Reads the data from `furniture.json` file.
  - [Completed] Handled by `module.load_metadata()` and `module.get_all_items()` on app startup.
- `filter_furniture()`: Takes all the furniture data and the user's filters, and returns a new list of items that match.
  - [Completed] Function logic exists in `module.py` and is ready to be integrated into a front-end search form
- `create_rental_order()`: Handles the checkout logic for rentals.
  - [Completed] Achieved via the Flask `checkout()` route, which calculates the total cost, generates a simulated `order_id`, and clears the cart.
- `create_c2c_listing()`: Handles the logic for a user posting a new item for sale.

🟦 Final Assessment

Biggest Change from the Initial Idea

The biggest change was to clearly split the project into two main sections: **"Official Rentals"** and a **"Public Marketplace."** This makes the business model feel more complete. I am also adding some "nice-to-have" features, like ”**AI Layout Advisor.”** This advisor will offer two distinct modes for users: 1) "Modern Design," focusing on aesthetics and space optimization, and 2) "Feng Shui," which provides advice based on traditional principles. This unique feature is designed to offer value beyond a simple transaction and serve as a differentiator from other platforms on the market.

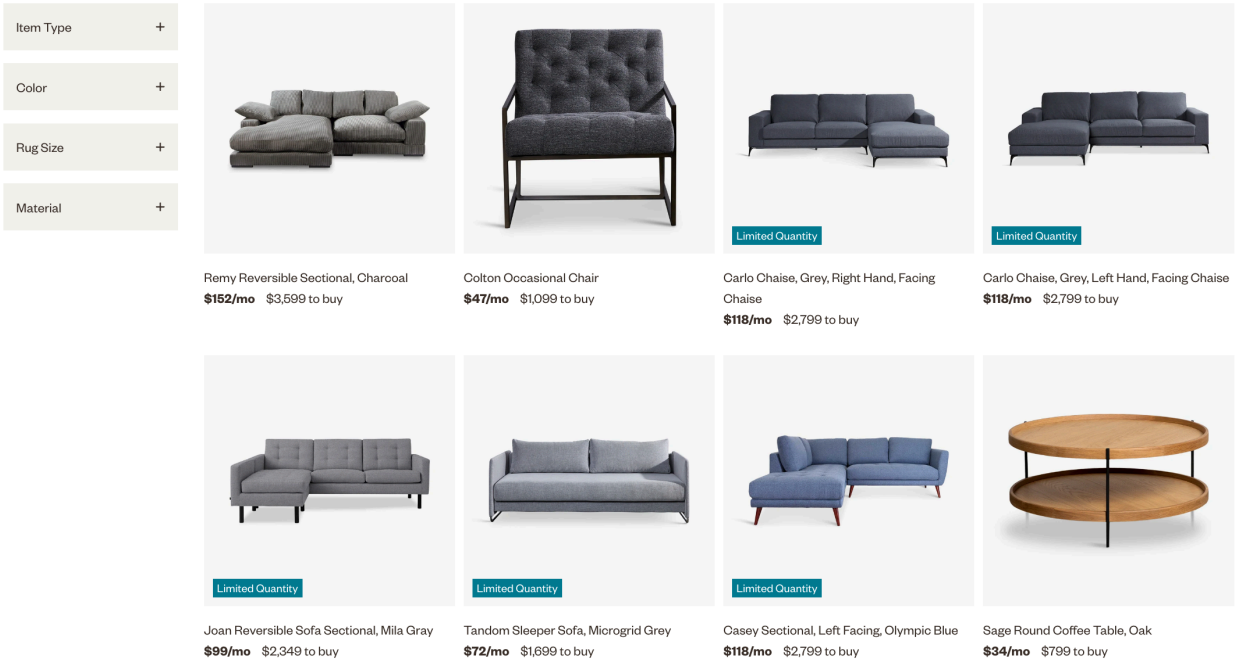
Confidence in Completing the Spec

After doing all the research, I feel excited but confident. 💪 The core rental and sales processes follow standard e-commerce logic. Although I have never built a project from scratch, I have experience querying databases, which will be very helpful for backend development. The biggest challenge will be the visual vibe on the front end.

Biggest Potential Problems

The main challenge is sourcing high-quality furniture data. As the professor pointed out, there are no readily available free furniture APIs, and the dataset on Kaggle is more like an inventory list and lacks visual appeal. The biggest initial hurdle will be creating a rich, simulated database similar to what is seen on fernish.com.

➡️ **Solution:** I can try to use an AI (like Gemini) to generate a batch of mock data in JSON format, complete with detailed descriptions and image URLs, to get the project started. Using simple web scraping techniques might be another solution.



Areas I'm Least Familiar With and Might Need Help

1. **Flask Application Architecture:** 0 experience building a website from scratch. Need help with the best way to structure the project's files and folders, plan the web page routes, and organize the code so it's easy to manage. I am really excited to embark!
2. **Frontend-Backend Communication:** The details of how to make the JavaScript on the frontend effectively talk to the Python/Flask backend is a key area to learn.

- **Database Creation and Connection:** While I know how to query a database, designing a schema from scratch and connecting it to Flask is a new challenge for me.
- **Frontend Development:** Creating a user-friendly interface with a sense of aesthetic will require a significant amount of time learning frontend technologies (JavaScript, CSS), and I may need more guidance in this area.
- **LLM API Integration:** Although the concept is straightforward, designing effective prompts and handling the technical details of the API integration are areas that will require exploration.

Future Work

ℹ️ AI Agent Feature Flow (Consider Feng Shui as an example.)

1. A user clicks a "Feng Shui Advice" button on their profile or a specific furniture page.
2. The Flask backend receives the request. It packages pre-defined Feng Shui rules (hard-coded rules like "the headboard of a bed should be against a solid wall" or "a desk should not face the door") along with the characteristics of the furniture in question (e.g., "this is a bed").
3. This packaged information is sent as a prompt to an LLM API. For example: "A user wants to place a bed. Based on the following Feng Shui rules {list of rules}, please provide 3 layout suggestions and explain how to use {recommended furniture types} to mitigate any negative energy."
4. The text-based advice returned from the LLM API is then displayed on the web page.

🧑🏻‍🦯 Story 2: Allen's Experience as a Seller

> Allen's friend gave him a floor lamp he doesn't need, so he decides to sell it on the "Marketplace" section of Modoya. After logging in, he clicks "Sell an Item." He inputs the item serial number, uploads a few photos of the lamp, fills in the title, description, and price, and chooses "Local Pickup Only."
>
> A couple hours later, a buyer contacts him through the site's messaging system, and they arrange a time to meet. After the transaction is complete, Allen goes back to the site, marks the item as "Sold," and leaves a positive review for the buyer.

🛠️ Technical Details:

- The platform must support C2C listings, allowing users to upload furniture photos, write descriptions, and set prices.
- A simple internal messaging system would be needed for buyers and sellers to communicate (this can be a goal for Version 2).

- Rental and purchase processes need to be handled separately but can share the same furniture database.