Review Doc

Nov. 3 Update

1. Response to Previous Feedback (Version 1 Review)

Following the feedback from the Version 1 review, the project has been fundamentally refactored to address the core requirement of building the application's "no-UI" logic first.

The primary action taken was the creation of `module.py`. This module now encapsulates all core business logic, completely independent of any user interface.

This module includes:

- **Data Management:** Functions to load all furniture data from the JSON files (`get_all_items`) and retrieve specific items by their ID (`get_item_by_id`).
- **Core Business Logic:**
  - `calculate_rent(item_metadata)`: Calculates the monthly rental price based on category, material, and style.
  - `calculate_buyout_price(item_metadata)`: Calculates the full buyout price.
  - `filter_furniture(...)`: A function to filter the item list based on criteria (this is ready to be implemented in the front-end).
- **No-UI Testing Functions:** The module also contains the `place_order` and `display_recommendations` functions, which use `input()` to test the complete order workflow in a terminal, as suggested in the feedback.

This refactoring directly addresses the previous feedback by establishing a testable, independent backend logic (`module.py`) before implementing the UI.

2. Current Implementation Status (Version 2)

With the `module.py` providing a solid foundation, a full-featured Flask web application (app.py) has been developed to serve as the Version 2 user interface.

The application successfully imports and utilizes the functions from `module.py`. The following features are fully implemented:

- **Product Catalog (Homepage):** The main page (`index.html`) dynamically loads all furniture items using `get_all_items()`. It correctly displays each item's image, series, style, and the prices calculated by `calculate_rent()` and `calculate_buyout_price()`.
- **Full Shopping Cart System:** A complete, session-based shopping cart is functional.
  - **Add to Cart:** Users can add items from the homepage directly to their cart.
  - **View Cart (`cart.html`):** This page displays all items in the session and calculates a subtotal and grand total.
  - **Update Cart Logic:** From the cart page, users can:
    1. **Update Duration:** Change the number of rental months. The total cost is recalculated instantly.
    2. **Toggle Order Type:** Switch an item between "RENT" and "BUY". The item's total cost updates accordingly (e.g., `monthly_rent * duration` or `buyout_price`).
    3. **Remove Item:** Remove items from the cart.
- **Checkout Process:**
  - The "Proceed to Checkout" button submits the final cart.
  - The `/checkout` route calculates the final total, generates a mock order ID, and clears the session cart.
  - It then renders a confirmation page (`checkout_complete.html`) showing the user their order ID and the total amount charged.

3. Next Steps

The current implementation provides a complete end-to-end user flow. The immediate next steps will focus on adding features and updating documentation:

1. **Implement Front-End Filtering:** The `module.py` file already contains the `filter_furniture` logic. The next task is to add UI elements (e.g., dropdown menus or forms) to `index.html` to allow users to filter the catalog by style, category, or color. This will involve updating the main `/` route in `app.py` to process the filter requests.
2. **Update Project Specification:** As requested in the previous review, the project specification document will be updated to reflect the new architecture (Flask, `module.py`, session management) and the specific features implemented in Version 2.

4. Blockers

There are no significant blockers at this time.

🟧 Overview

Modoya is a furniture platform designed for young, style-conscious people on a budget, as well as anyone who cares about sustainability. It addresses the problem of "fast furniture" waste often seen in college towns. Modoya aims to solve this with two core features: "Official Designer Rentals" and a "Public Second-hand Marketplace", making a stylish lifestyle more accessible while promoting environmental sustainability.

🟦 Implemented Part

My progress has been focused on the project's infrastructure and core data preparation.

- **Project Architecture:** Set up the basic Python Flask project structure and virtual environment.
- **API Connection:** Successfully authenticated and fixed issues with the OpenAI API for image generation.
- **Database Structure Design:** To generate high-quality AI images, I researched and refined the `furniture_table_with_images.csv` data structure, adding richer columns like `series` (e.g., "Eames Lounge Chair") and `style` (e.g., "Bauhaus style").

- **Prompt Engineering:** Iteratively tested and refined the prompt in the Python script to produce photorealistic "commercial product photography" instead of "clay models."
- **Data Automation Script:** Wrote `generate_csv_data.py` to randomly generate batch features for the furniture, preparing for mass image generation.

🟩 Demonstration

As the front-end has not been started, my demonstration focuses on the data preparation pipeline:

A. Data Structure (CSV Example):

category	series	style	material	color
Bed Frame	AIDENGREENQ by Meridian - Bed	Chrome Nailheads, Wing Design and Contemporary Style	Velvet Upholstery	Green
Sofa	Italian Designer Leather Button Upholstered Sofa	A touch of sophistication and opulence with sumptuous curves which create the most striking outline	covered with a warm fudge brown high quality Italian leather with velvet covered	Beige

B. Image Generation Results (From Prompt Iteration):

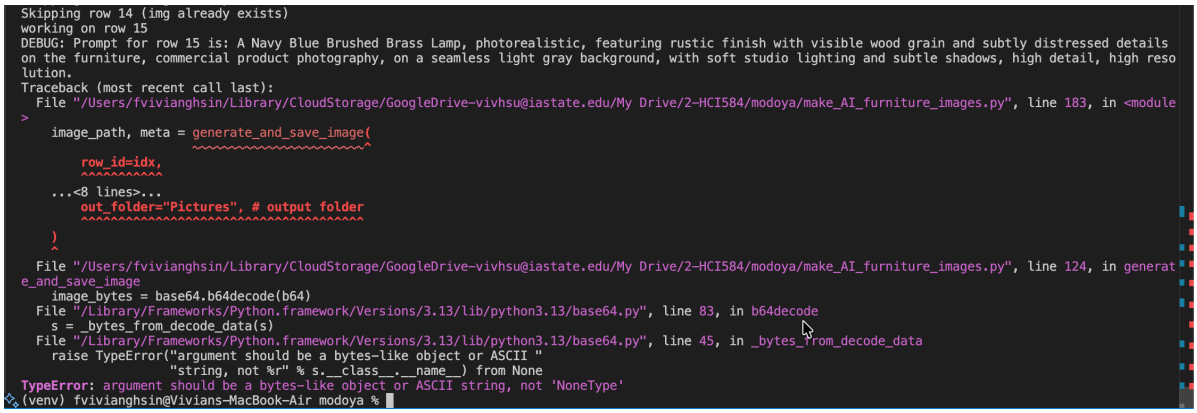
🟨 Issues Encountered

Solved Issues:

- **Problem:** Initial AI-generated images did not look real.
- **Solution:** I realized this was a prompt engineering problem. By adding more descriptive columns to the CSV (like `series`, `style`) and refining the prompt string to `f"A {color} {material} {full_category}, {final_style}{extras_part}, commercial product photography..."`, I significantly improved the photorealism of the images.

Unsolved Issues (Current Blocker):

- **Problem Description:**
  After successfully writing `generate_csv_data.py` to generate a batch of data (e.g., 100 rows), my main image generation script (`make_AI_furniture_images.py`) **gets stuck** when I try to run it on this new batch file.
- **Specific Error:**



- **Current Status:**
  I have temporarily paused debugging this bug because I am worried about falling behind schedule and wanted to start on the website front-end. However, I have not yet started the front-end development.

🟩 Current **Solutions**

I realize that even though I want to start the website, I cannot move forward until the core data (image) batch generation problem is solved.

Therefore, my proposed plan is:

- Prioritize Debugging: My first priority this week is to solve the "stuck" bug. I will carefully check the CSV format from generate_csv_data.py to ensure it perfectly matches the logic in my main script.
- I might need the professor's help. 🙏

🟦 Milestones for the Next Weeks

My V2 goal is: **Complete a database of at least 500 furniture images and build a working website prototype with core browsing and (simulated) ordering functionality.**

To achieve this, I have set 5 milestones:

1. **Milestone 1: The Unblocker (Core Database)**
   - **Description:** Fix the current batch-generation bug. Successfully generate and manually curate a database of at least 500 high-quality furniture images with corresponding CSV data.

2. **Milestone 2: Build Flask Backend & Data Logic**
    - **Description:** Set up the basic routes for the Flask application. Implement backend core functions like `load_furniture_data()` and `filter_furniture()` to read and filter the furniture data.
3. **Milestone 3: Implement Frontend Browsing & Filtering (Story 1 - Part 1)**
    - **Description:** Build the HTML/CSS/JS for the product list page. Implement the front-end filters (style, price range) and make them communicate with the Flask backend to update the page.
4. **Milestone 4: Implement Core Rental Workflow (Story 1 - Part 2)**
    - **Description:** Build the product detail page, shopping cart , and a simulated checkout page. Implement the backend `create_rental_order()` to (simulate) an order and have it appear in a user's account.
5. **Milestone 5: Build User Account Prototype**
    - **Description:** Create a basic user account page where a user (simulated login) can see their current rentals and rental history, as outlined in the spec.

🟪 Self-Reflection

- **Are you satisfied with your progress so far?**

  Honestly, I am a bit behind my original schedule. This is mostly because I only have time on weekends, and even that time has been limited recently. As we move into the second half of the semester, I hope to have found a better balance and plan to dedicate more time here.

- **Do you think you can finish the project within the next weeks?**

  I believe it's possible, but it is conditional on me strictly following a new goal I've set for myself: to spend at least three half-days, totaling ~18 hours per week, on this project.

- **Expectations vs. reality: what was easier, what was more difficult?**
    - *More difficult:* The AI image generation adjustment. This took much, much longer than I expected. The process feels like a "black box" for me because I'm responsible for feeding it prompts and seeing the results, but I

can't control the process in between. It makes me a little bit anxious (is this the price of using AI?). If I wasn't so insist about the image quality, I could have generated thousands of images by now, but I feel the visual quality is worth the effort.

    - *Easier:* Writing the `generate_csv_data.py` script to automate random feature generation was more straightforward than I thought.

- **Any light bulb moments?**
    - While repeatedly tweaking the prompts, I truly understood the challenge I mentioned in my spec. I realized this project's success is highly dependent on the quality of the "rich, simulated database". Without a visually appealing dataset, even the best front-end work will look cheap. This solidified my decision to spend the necessary time to get the data right first.