Project1 - Flora

Problem Description:

In Karnataka, India, small floriculture businesses are a key source of income for many households. Every day, a variety of products such as White Jasmine, Pink Jasmine, Kakda Jasmine, and Crossandra are harvested, bundled, and sold to intermediaries. These intermediaries play a role in transporting the flowers through a chain of middlemen before they eventually reach the flower market.

Issues:

1. **Lack of Expense Tracking:** Many floriculture businesses struggle with profitability due to the absence of an efficient expense tracking system. Expenses for pesticides, fertilizers (like NPK), labor, and other inputs are often unmonitored, resulting in higher costs and limited profits.
2. **No Access to Real-Time Base Market Pricing:** The Karnataka floriculture authority sets the daily base market price of flowers at noon. However, many florists don't receive this critical information until the next day through newspapers, WhatsApp groups, or word of mouth.
3. **Lengthy Supply Chain:** A complex supply chain involving multiple middlemen not only increases the time it takes for flowers to reach the market but also impacts the quality of the product. Additionally, farmers' earnings are reduced due to commissions taken at each step of the supply chain.

Rules:

- Business owners should be able to login to website and input their expense.
- Owners should be able to access historical data.
- Export feature should be available to export data into reports (Pdf, Excel etc)
- Business owners should be able to track daily, weekly, monthly sales.
- Provide the ability for businesses to track the number of flowers in stock.
- Notify businesses when stock is low on certain products.
- Sellers should be able to view and manage customer details, purchase history, and preferences
- Customers should be able to search products by category, price, and availability.
- Use geolocation to show nearby flower vendors for faster delivery or pickup.
- Farmers should be able to view the market value of flowers set by Floriculture authority on a given day.
- Customers should not be connected to the vendors that dont have stock available.
- The selling price of a unit product should not be below market base price.
- 4 garlands are bundled into one unit.
- Ensure that orders greater than stock in hand are not accepted. For orders that are received and not shipped, requested quantity should be reserved.
- Customers should be able to place orders online.

Nouns:

- Products
- White Jasmine, Pink Jasmine, Crossandra
- Intermediaries
- Expense
- Pesticides
- Fertilizer
- Labor
- Customers
- Farmers
- Daily base market Price
- Order

Verbs:

- Bundled
- Sold
- Monitored
- Export
- Track
- Manage
- Search
- View
- Reserve

Categorize Nouns:

- Products:
    - Name: {White Jasmine, Pink Jasmine, Crossandra}

- Expense:
    - Category: {Machinery and Tools, One time Installation, Fertilizers and Soil Amendments, Pesticides and Herbicides, Labor Cost, Water and Electricity}
    - Amount
    - Date

- Customer:
    - Name
    - Email
    - Contact
    - Address
    - Latitude
    - Longitude

- Farmer:
    - Name
    - Email
    - Contact
    - Address
    - Latitude
    - Longitude

- Order:
    - Product
    - Qty
    - Amount

- Base Market Price
    - Date
    - Price
    - Product

Functionality picked to be part of Redis

- A unique session should be created for a given user.
- Session should be active for 30 mins
- Users who are authenticated should not be asked to authenticate until and unless they explicitly logout.

Redis Implementation:

Session Management:

- You've integrated `connect-redis` with `express-session` to store session data in Redis.
- This setup ensures persistent session management, where user sessions are stored in Redis instead of in-memory, enhancing scalability.

Redis Client Setup:

- You configured a Redis client using `redis` npm package with `legacyMode` enabled for compatibility with `connect-redis`.
- The Redis client is connected using the URL `redis://localhost:6379`, and you handled connection errors by logging them.

Session Configuration:

- The session is configured with:
  - A `RedisStore` backed by your Redis client.
  - A `secret` key for session encryption (you later planned to store it securely in an environment variable).
  - A session timeout (`maxAge`) set to `30 minutes` or dynamically updated as needed.

Session Expiry Handling:

- You display the remaining session time on the `sellerOptions` page using a JavaScript timer that calculates the time until the session expires.
- When the session expires, the user is automatically redirected to the `/logout` route.

Logout Functionality:

- The logout functionality destroys the Redis-backed session and clears the session cookie (`connect.sid`) before redirecting users to the home page.

Conditional Display Logic:

- You conditionally display the logout button on the navigation bar using the `showLogout` variable, ensuring it's only visible on specific pages like `sellerOptions`.

<u>Redis DataStructures used in the Project.</u>

1) Session Management

- **Data Structure**: **Redis String** (via `connect-redis` and `express-session`).
- **Key**: `"sess:<sessionID>"`.
- **Value**: A serialized JSON object containing:
  - `cookie`: Information about the session cookie (e.g., `maxAge`, `expires`, etc.).
  - `seller`: An object storing session data for the logged-in seller:
    - `id`: Seller's unique identifier.
    - `email`: Seller's email address.
    - `name`: Seller's full name.
- **Purpose**:
  - Securely manages user sessions by persisting data in Redis.
  - Provides high-performance session storage for seamless user authentication.
  - Automatically handles session expiration using Redis's TTL (Time-to-Live).

2) Real-time Session Timer

- **Data Structure**: **TTL (Time-to-Live)**, a feature of Redis.
- **Key**: `"sess:<sessionID>"`.
- **Purpose**:
    - Tracks session expiration based on the `maxAge` configured in session cookies.
    - Used in the UI to display a countdown timer for the remaining session time.
    - Automatically deletes session data after expiration, ensuring efficient resource usage.