**WDD 330 Final Web Application Proposal**

**Project Proposal: “FBI Watchlist”**

**Overview**

FBI Watchlist is a web application that lets users view and track individuals from the FBI’s Wanted list. It solves the problem of easily accessing and organizing information about wanted persons by fetching data from the FBI Wanted API and allowing users to save details locally for later reference. I’m creating this because I’m interested in public safety data and want a simple tool to explore it.

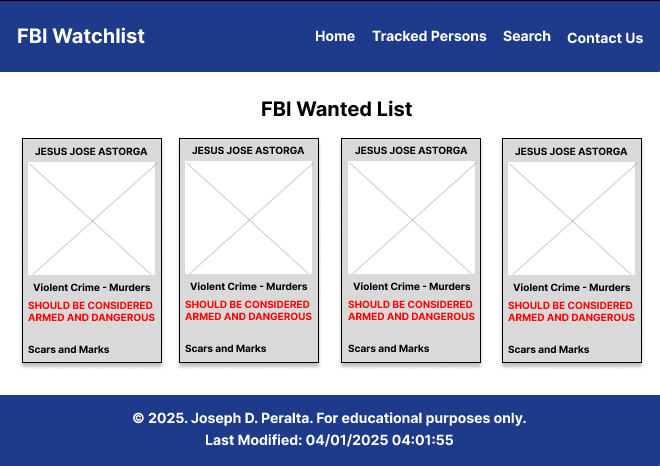
**Target Audience**

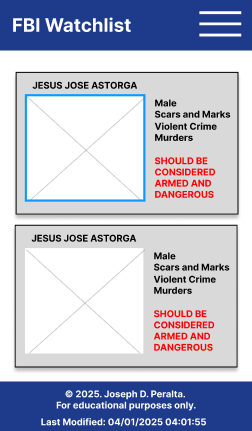
• People curious about crime and public safety updates to serve them public awareness and help in public safety.

**Major Functions**

1. **Search Wanted List:** Users enter a keyword (e.g., "Murder") to filter the FBI Wanted list.
2. **Display Results:** Shows a list of wanted persons with name, image, and crime description.
3. **Track Wanted Persons:** Users can click a button to store a wanted person’s details in Local Storage for later viewing.
4. **View Tracked List:** A separate page shows all the wanted persons they’ve chosen to track.
5. **Basic Design:** Clean, mobile-friendly layout.
6. **Clear Tracked List**: Allows users to remove all tracked persons from Local Storage with one action. Adds a "Clear All" button on the Tracked List Page that calls localStorage.clear() or resets the tracked key, refreshing the page to show an empty state.
7. **Show Additional Details**: Displays extra information about a wanted person on demand(e.g. caution or warning)
8. **Sort Results**: Let users sort the displayed results by name or crime type.

**Wireframes**

****

****

**External Data**

* **FBI Wanted API:** Free public API (e.g., https://api.fbi.gov/wanted/v1/list) to fetch data like names, images, and crime details.

**Data Storage**  
• **Local Storage:** Saves tracked persons as JSON (e.g., {"title": "JESUS JOSE ASTORGA", "description": "Murder"}).

**Module List**

**UI Module**

* Handles the layout and display of the search form, results, and tracked list page.

**API Module**

* Fetches and retrieves data from the FBI Wanted API.

**Search Module**

* Filters API data based on user-entered keywords.

**Tracking Module**

* Manages saving and loading tracked persons in Local Storage.

**Data Management Module**

* Processes and cleans API data for consistent use.

**Initial Files**

* index.html, tracked.html, styles.css, app.js, placeholder.json, images.webp

**Graphic Identity**

• **Color Scheme:** Dark blue (#1E3A8A) and red (#DC2626) for an authoritative feel.  
• **Typography:** Default sans-serif font (e.g., Arial).  
• **Icon:** A basic pin symbol (📍) for tracking (CSS or emoji).

**Timeline**

• **Week 5:**

* Create HTML for search form and results section.
* Style with CSS for a clean layout.
* **Week 6:**
* Add JavaScript to fetch and display FBI API data.
* Implement keyword filtering.
* **Week 7:**
* Add track-to-Local-Storage and tracked list page functionality.
* Test and finalize.

**Sample Workflow**

1. User enters "Murder" in the search box and clicks "Search."
2. App fetches data from https://api.fbi.gov/wanted/v1/list and filters for "Murder" in subjects or description.
3. Results show (e.g., "JESUS JOSE ASTORGA - Murder" with photo).
4. User clicks "Track" to store it in Local Storage.
5. Tracked list page displays saved entries like "JESUS JOSE ASTORGA."

**Project Planning**

[**https://trello.com/invite/b/67eb98ec1f8d9d36f1a3ea78/ATTI8a82539d0734f642d5864d1c3eb2dd8dC295398A/wdd-330-final-project**](https://trello.com/invite/b/67eb98ec1f8d9d36f1a3ea78/ATTI8a82539d0734f642d5864d1c3eb2dd8dC295398A/wdd-330-final-project)

**Challenges**

**Handling API Data Variability**

* **Description:** The FBI Wanted API returns a lot of data (e.g., names, images, descriptions), but not all fields are consistently populated. For example, some entries might lack images, description, or subjects, which could break the display or filtering logic if not handled properly.
* **Impact:** Users might see incomplete or broken results (e.g., no photo or missing crime details), making the app less reliable.
* **Solution:** Add fallback checks in JavaScript (e.g., item.images[0]?.thumb || 'placeholder.jpg' for missing images) and display "N/A" for missing fields. Test with varied API responses to ensure robustness.