CIS 3100 - Management Information Systems

Core Workflow

- Use a large language model based integrated development environment such as Google Gemini to develop a core workflow.

Goal:

- ➢ Collect all selected genres, tags, and fandoms (which are marked with an active CSS class).
- ➢ Generate a single, clean URL query string that represents the user's choices.
- ➢ Simulate navigating to that unique search result URL.

Code (Developed by Google Gemini):

```
<!DOCTYPE html>
<html lang="en" class="light">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Advanced Novel Search Interface</title>
    <!-- Load Tailwind CSS and configure Dark Mode -->
    <script src="https://cdn.tailwindcss.com"></script>
    <script>
        // Configure Tailwind to use the 'dark' class on the HTML element for
dark mode
        tailwind.config = {
            darkMode: 'class',
            theme: {
                extend: {
                    fontFamily: {
                        sans: ['Inter', 'sans-serif'],
                    },
                    colors: {
                        'primary-light': '#4f46e5', // Indigo-600
                        'primary-dark': '#6366f1',  // Indigo-500
                    }
                },
            },
        }
    </script>
```

```
    <!-- Use Inter font and set base styles -->
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Inter:wght@100..900&display=swap'
);
        body {
            font-family: 'Inter', sans-serif;
            background-color: #f1f5f9; /* Light Slate Background */
            transition: background-color 0.3s;
        }
        .dark body {
            background-color: #1f2937; /* Dark Gray Background */
        }
        /* Custom styles for smoother transition on collapse */
        .collapse-content {
            max-height: 0;
            overflow: hidden;
            transition: max-height 0.4s ease-in-out;
        }
        .collapse-content.active {
            /* Set a height that is guaranteed to be taller than the content */
            max-height: 2000px;
        }
        /* Ensure the body transition works across themes */
        .dark .bg-gray-100 { background-color: #374151; }
        .dark .text-gray-700 { color: #d1d5db; }
    </style>
</head>
<body class="p-4 sm:p-8">

    <div class="max-w-4xl mx-auto">

        <!-- Header -->
        <header class="flex justify-between items-center mb-6 p-4 bg-white
dark:bg-gray-800 rounded-xl shadow-lg">
            <h1 class="text-3xl font-extrabold text-indigo-600
dark:text-indigo-400">Advanced Novel Search</h1>
```

```
            <!-- Theme Toggle -->
            <button id="theme-toggle" class="p-2 rounded-full bg-gray-100
dark:bg-gray-700 text-gray-700 dark:text-gray-300 hover:ring-2 ring-indigo-500
transition-colors duration-200" title="Toggle theme">
                <span id="theme-icon"></span>
            </button>
        </header>

        <!-- Main Search Form -->
        <main class="bg-white dark:bg-gray-800 p-6 sm:p-8 rounded-xl shadow-2xl
space-y-8">

            <!-- Basic Search Fields -->
            <section class="space-y-4">
                <h2 class="text-xl font-semibold text-gray-800
dark:text-gray-100 border-b pb-2 border-gray-200 dark:border-gray-700">Basic
Search</h2>

                <!-- Title Input -->
                <div>
                    <label for="novel-title" class="block text-sm font-medium
text-gray-700 dark:text-gray-300 mb-1">Novel Title</label>
                    <input type="text" id="novel-title" class="w-full p-3
border border-gray-300 dark:border-gray-600 rounded-lg bg-gray-50
dark:bg-gray-700 text-gray-900 dark:text-gray-100 focus:ring-indigo-500
focus:border-indigo-500 transition-colors" placeholder="e.g., The Secret of the
Ancient Library">
                </div>

                <!-- Author Input -->
                <div>
                    <label for="author-input" class="block text-sm font-medium
text-gray-700 dark:text-gray-300 mb-1">Author</label>
                    <input type="text" id="author-input" class="w-full p-3
border border-gray-300 dark:border-gray-600 rounded-lg bg-gray-50
dark:bg-gray-700 text-gray-900 dark:text-gray-100 focus:ring-indigo-500
focus:border-indigo-500 transition-colors" placeholder="e.g., J. M. Richards">
                </div>
```

```
                <!-- Summary/Description Input -->
                <div>
                    <label for="summary-input" class="block text-sm font-medium
text-gray-700 dark:text-gray-300 mb-1">Summary Keywords</label>
                    <textarea id="summary-input" rows="3" class="w-full p-3
border border-gray-300 dark:border-gray-600 rounded-lg bg-gray-50
dark:bg-gray-700 text-gray-900 dark:text-gray-100 focus:ring-indigo-500
focus:border-indigo-500 transition-colors" placeholder="Keywords to search in
the novel's description..."></textarea>
                </div>
            </section>

            <!-- Genre/Tag/Fandom Filters (Collapsible) -->
            <section class="space-y-4">
                <h2 class="text-xl font-semibold text-gray-800
dark:text-gray-100 border-b pb-2 border-gray-200 dark:border-gray-700">Filters
& Sorting</h2>

                <!-- Collapse Toggle Button -->
                <button onclick="toggleCollapse('advanced-filters')"
data-target="advanced-filters" class="w-full text-left flex justify-between
items-center p-3 rounded-lg bg-indigo-50 dark:bg-indigo-900 text-indigo-700
dark:text-indigo-200 font-semibold hover:bg-indigo-100 dark:hover:bg-indigo-800
transition-colors">
                    Advanced Filters (Genres, Tags, Fandoms)
                    <svg class="w-5 h-5 transition-transform duration-300"
xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M19 9l-7 7-7-7"></path></svg>
                </button>

                <!-- Collapsible Content -->
                <div id="advanced-filters" class="collapse-content space-y-6
p-4 border border-gray-200 dark:border-gray-700 rounded-lg bg-gray-50
dark:bg-gray-700">

                    <!-- Genre Selection -->
```

```
                    <div>
                        <h3 class="text-lg font-medium text-gray-800
dark:text-gray-100 mb-2">Genres (Click to Include, Double-Click to
Exclude)</h3>
                        <div id="genre-buttons" class="flex flex-wrap">
                            <!-- Genre buttons will be injected here -->
                        </div>
                    </div>

                    <!-- Tag Selection -->
                    <div>
                        <h3 class="text-lg font-medium text-gray-800
dark:text-gray-100 mb-2">Tags</h3>
                        <div id="tag-buttons" class="flex flex-wrap">
                            <!-- Tag buttons will be injected here -->
                        </div>
                    </div>

                    <!-- Fandom Selection -->
                    <div>
                        <h3 class="text-lg font-medium text-gray-800
dark:text-gray-100 mb-2">Fandoms</h3>
                        <div id="fandom-buttons" class="flex flex-wrap">
                            <!-- Fandom buttons will be injected here -->
                        </div>
                    </div>
                </div>

                <!-- Sorting Controls -->
                <div class="flex flex-col sm:flex-row space-y-4 sm:space-y-0
sm:space-x-4 items-end pt-4">

                    <!-- Sort By Selector -->
                    <div class="w-full sm:w-1/2">
                        <label for="sort-by" class="block text-sm font-medium
text-gray-700 dark:text-gray-300 mb-1">Sort By</label>
                        <select id="sort-by" class="w-full p-3 border
border-gray-300 dark:border-gray-600 rounded-lg bg-gray-50 dark:bg-gray-700
```

```
text-gray-900 dark:text-gray-100 focus:ring-indigo-500
focus:border-indigo-500">
                            <option value="relevance">Relevance</option>
                            <option value="last_updated">Last Updated</option>
                            <option value="rating">Rating</option>
                            <option value="chapters">Chapter Count</option>
                        </select>
                    </div>

                    <!-- Order Toggle -->
                    <div class="w-full sm:w-1/2">
                        <label class="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">Order</label>
                        <div id="order-toggle" class="flex rounded-lg
overflow-hidden border border-gray-300 dark:border-gray-600">
                            <button data-order="asc" onclick="setOrder(this)"
class="flex-1 py-3 text-sm font-medium transition-colors duration-200">
                                <span class="sr-only">Ascending</span>
                                <!-- Asc Icon -->
                                <svg class="w-5 h-5 mx-auto"
xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
stroke="currentColor" stroke-width="2"><path stroke-linecap="round"
stroke-linejoin="round" d="M3 4h18M3 8h18m-7 4h7m-7 4h7M7 12V4m0
8l-3-3m3 3l3-3"></path></svg>
                            </button>
                            <button data-order="desc" onclick="setOrder(this)"
class="flex-1 py-3 text-sm font-medium transition-colors duration-200">
                                <span class="sr-only">Descending</span>
                                <!-- Desc Icon -->
                                <svg class="w-5 h-5 mx-auto"
xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
stroke="currentColor" stroke-width="2"><path stroke-linecap="round"
stroke-linejoin="round" d="M3 4h18M3 8h18m-7 4h7m-7 4h7m-7 0V12m0
8l3-3m-3 3l-3-3"></path></svg>
                            </button>
                        </div>
                    </div>
                </div>
```

```
            </section>

            <!-- Search Button -->
            <button onclick="executeSearch()" class="w-full py-4 bg-indigo-600
hover:bg-indigo-700 dark:bg-indigo-500 dark:hover:bg-indigo-400 text-white
font-bold text-lg rounded-xl shadow-lg transition-colors duration-300
focus:outline-none focus:ring-4 focus:ring-indigo-500/50">
                Execute Advanced Search
            </button>

            <!-- Console Output Area -->
            <div id="output-log" class="p-4 bg-gray-900 dark:bg-gray-950
text-green-400 font-mono text-sm rounded-lg whitespace-pre-wrap">
                > Search output will appear here after clicking "Execute
Advanced Search".
            </div>

        </main>
    </div>

    <script>
        // ======================================================================
        // MOCK DATA AND UTILITY FUNCTIONS
        // ======================================================================

        // Mock Data (required for initialization)
        const GENRES = ['Fantasy', 'Romance', 'Sci-Fi', 'Thriller', 'Horror',
'Historical'];
        const TAGS = ['Slice of Life', 'Adventure', 'Action', 'Mystery',
'Dark', 'Harem', 'Gore'];
        const FANDOMS = ['Original', 'HP', 'Marvel', 'Star Wars', 'Lord of the
Rings'];

        // Mock Icons (required for initialization)
        const sunIcon = `<svg xmlns="http://www.w3.org/2000/svg" width="24"
height="24" viewBox="0 0 24 24" stroke="currentColor" stroke-width="2"
fill="none" stroke-linecap="round" stroke-linejoin="round" class="w-5
```

```
h-5"><circle cx="12" cy="12" r="4"></circle><path d="M12 2v2"></path><path
d="M12 20v2"></path><path d="M4.93 4.93l1.41 1.41"></path><path d="M17.66
17.66l1.41 1.41"></path><path d="M2 12h2"></path><path d="M20
12h2"></path><path d="M4.93 19.07l1.41-1.41"></path><path d="M17.66
6.34l1.41-1.41"></path></svg>`;
        const moonIcon = `<svg xmlns="http://www.w3.org/2000/svg" width="24"
height="24" viewBox="0 0 24 24" stroke="currentColor" stroke-width="2"
fill="none" stroke-linecap="round" stroke-linejoin="round" class="w-5
h-5"><path d="M21 12.79A9 9 0 1 1 11.21 3 7 7 0 0 0 21 12.79z"></path></svg>`;

        // Utility Functions (required for initialization)

        // Function to safely log to the dedicated UI area
        function logToConsole(message) {
            const logElement = document.getElementById('output-log');
            if (logElement) {
                logElement.textContent += `\n> ${message}`;
                logElement.scrollTop = logElement.scrollHeight; // Auto-scroll
            }
        }

        function toggleTheme() {
            const html = document.documentElement;
            const isDark = html.classList.toggle('dark');
            localStorage.setItem('theme', isDark ? 'dark' : 'light');

            const icon = document.getElementById('theme-icon');
            if (isDark) {
                icon.innerHTML = moonIcon;
            } else {
                icon.innerHTML = sunIcon;
            }
        }

        function getButtonClasses(state) {
            // Tailwind classes for buttons based on state
            const base = 'px-3 py-1 rounded-full text-sm font-medium
transition-all duration-200 m-1 whitespace-nowrap focus:outline-none
```

```
focus:ring-2 ';

            switch (state) {
                case '1': // Include (Green)
                    return base + 'bg-green-100 text-green-700
dark:bg-green-700 dark:text-green-100 ring-green-500 hover:bg-green-200
dark:hover:bg-green-600';
                case '2': // Exclude (Red)
                    return base + 'bg-red-100 text-red-700 dark:bg-red-700
dark:text-red-100 ring-red-500 hover:bg-red-200 dark:hover:bg-red-600';
                case '0': // Neutral (Gray)
                default:
                    return base + 'bg-gray-200 text-gray-700 dark:bg-gray-700
dark:text-gray-300 ring-indigo-500 hover:bg-gray-300 dark:hover:bg-gray-600';
            }
        }

        function createButtons(containerId, dataArray) {
            const container = document.getElementById(containerId);
            if (!container) return;

            dataArray.forEach(value => {
                const btn = document.createElement('button');
                btn.textContent = value;
                btn.dataset.value = value;
                btn.dataset.state = '0'; // 0: Neutral, 1: Include, 2: Exclude
                btn.className = getButtonClasses('0');
                btn.addEventListener('click', function() {
                    let state = parseInt(this.dataset.state);
                    state = (state + 1) % 3; // Cycle: 0 -> 1 -> 2 -> 0
                    this.dataset.state = state.toString();
                    this.className = getButtonClasses(state.toString());
                });
                container.appendChild(btn);
            });
        }

        function setOrder(clickedButton) {
```

```
            const parent = document.getElementById('order-toggle');
            if (!parent) return;

            const buttons = parent.querySelectorAll('button');
            const activeClasses = ['bg-indigo-500', 'text-white',
'dark:bg-indigo-600', 'shadow-md'];
            const inactiveClasses = ['bg-gray-200', 'text-gray-700',
'dark:bg-gray-700', 'dark:text-gray-300', 'hover:bg-gray-300',
'dark:hover:bg-gray-600'];

            buttons.forEach(btn => {
                btn.classList.remove(...activeClasses);
                btn.classList.add(...inactiveClasses);
            });

            clickedButton.classList.remove(...inactiveClasses);
            clickedButton.classList.add(...activeClasses);
        }

        function toggleCollapse(id) {
            const content = document.getElementById(id);
            const button = document.querySelector(`[data-target="${id}"]`);
            if (content) {
                content.classList.toggle('active');
                const isActive = content.classList.contains('active');
                content.style.maxHeight = isActive ? content.scrollHeight +
"px" : "0";
                // Simple arrow rotation in the button
                const arrow = button.querySelector('svg');
                if (arrow) {
                    arrow.classList.toggle('rotate-180', isActive);
                }
            }
        }

        // ================================================================
        // CORE WORKFLOW: executeSearch() (Developed via LLM)
        // This function collects all filters and generates the search query
```

```
URL.
        // ====================================================================
        function executeSearch() {
            logToConsole("--- Advanced Search Executed ---");
            let queryParams = [];

            // 1. Capture Text Inputs (Title, Author, Summary)
            const title = document.getElementById('novel-title').value.trim();
            const author =
document.getElementById('author-input').value.trim();
            const summary =
document.getElementById('summary-input').value.trim();

            if (title) queryParams.push(`title=${encodeURIComponent(title)}`);
            if (author)
queryParams.push(`author=${encodeURIComponent(author)}`);
            if (summary)
queryParams.push(`summary=${encodeURIComponent(summary)}`);

            // 2. Capture Tag/Genre/Fandom Buttons (State 1: Include, State 2:
Exclude)
            const tagGroups = [
                { selector: '#genre-buttons', param: 'genre' },
                { selector: '#tag-buttons', param: 'tag' },
                { selector: '#fandom-buttons', param: 'fandom' }
            ];

            tagGroups.forEach(group => {
                // Find all buttons that are NOT in the neutral (0) state
                const activeButtons = document.querySelector(group.selector)
                    .querySelectorAll('button[data-state="1"],
button[data-state="2"]');

                if (activeButtons.length > 0) {
                    const includeValues = [];
                    const excludeValues = [];

                    activeButtons.forEach(btn => {
```

```
                        const value = btn.dataset.value;
                        if (btn.dataset.state === '1') {
                            includeValues.push(value); // State '1' is INCLUDE
                        } else if (btn.dataset.state === '2') {
                            excludeValues.push(value); // State '2' is EXCLUDE
                        }
                    });

                    // Add Include parameters (e.g., genre=Fantasy,Romance)
                    if (includeValues.length > 0) {

queryParams.push(`${group.param}=${includeValues.map(encodeURIComponent).join('
,')}`);
                    }
                    // Add Exclude parameters (e.g., exclude_tag=Harem)
                    if (excludeValues.length > 0) {

queryParams.push(`exclude_${group.param}=${excludeValues.map(encodeURIComponent
).join(',')}`);
                    }
                }
            });

            // 3. Capture Sorting and Order
            const sortBy = document.getElementById('sort-by').value;
            // The logic below looks for the currently active button in the
order toggle group
            const orderElement = document.querySelector('#order-toggle
.bg-indigo-500, #order-toggle .dark\\:bg-indigo-600');
            const order = orderElement ?
orderElement.getAttribute('data-order') : 'asc'; // Default to asc if not found

            queryParams.push(`sort_by=${sortBy}`);
            queryParams.push(`order=${order}`);


            // 4. Construct the Final URL
            const baseUrl = 'search_results.html';
```

```
            let finalUrl = baseUrl;
            if (queryParams.length > 0) {
                // Join all parameters with '&' and prepend with '?'
                finalUrl += '?' + queryParams.join('&');
            }

            // 5. Output the result for the assignment demonstration
            console.log("Simulated Query URL:", finalUrl); // Also log to
browser console
            logToConsole(`Simulated Query URL: ${finalUrl}`);

            // In a live app, you would use:
            // window.location.href = finalUrl;
        }
        // ================================================================


        // --- Initialization ---

        document.addEventListener('DOMContentLoaded', () => {
            // Clear initial console text
            document.getElementById('output-log').textContent = "> Ready to
search...";

            // 1. Initialize filter buttons
            createButtons('genre-buttons', GENRES);
            createButtons('tag-buttons', TAGS);
            createButtons('fandom-buttons', FANDOMS); // Initialize new Fandom
buttons

            // 2. Initialize theme toggle and set initial icon
            const themeToggle = document.getElementById('theme-toggle');
            const icon = document.getElementById('theme-icon');
            themeToggle.addEventListener('click', toggleTheme);

            // Check for stored theme preference
            const isDark = localStorage.getItem('theme') === 'dark';
            if (isDark) {
```

```
                document.documentElement.classList.add('dark');
                icon.innerHTML = moonIcon;
            } else {
                icon.innerHTML = sunIcon;
            }

            // 3. Initialize order button (default to Ascending)
            const initialOrderButton = document.querySelector('#order-toggle
[data-order="asc"]');
            if (initialOrderButton) {
                setOrder(initialOrderButton);
            }
        });

    </script>
</body>
</html>
```

LINK:

https://gitschoolhub.github.io/CIS-3100---Advanced-Novel-Search/src/index.html

You can see this code play out in the UI/UX, just play around with the search bars or the interactive buttons and dropdowns. Then hit the search button at the bottom. A url will pop up (It does nothing) but you can understand the general idea that for this website, the url will redirect you allowing you to narrow your search results and better find the novel that you are looking for.

Of course this website will not be able to work since I don't have a vast catalogue or database of novels to search through, nor one that is properly categorized into different subtexts, so until I do this, it will not be able to fully function.

One way I could achieve this affect without having to go through the tedious task of uploading information about every novel from different websites is through data mining or building a community fanbase that will help me to do so. But this will take a bunch of time and there is no guarantee that this will occur. The easiest way would be to get an API from each website to better navigate their novels using the advanced search that I created. Sadly though, most novel websites don't create APIs for their site, mostly because of privacy issues or because they just can't be bothered to do so. But there are some unofficial APIs that I can maybe use, which would be nice.

Technical Product

- Deploy the workflow to the cloud fireplace and iterate the technical product.

Here is the Link to my Firebase (Preview):
https://9000-firebase-studio-1761759525118.cluster-pb4ljhlmg5hqsxnzpc56r3prxw.cloudworkstations.dev

The link will also be in my GitHub README.

Notes:

This works pretty similarly to the UI/UX that I have created with Gemini earlier in Subset 2. Because my account is not verified to be 18+, I wasn't able to get an API key from Gemini to fully utilize the search engine.

Despite this drawback, I did learn some key points from this Firebase App.

1. I will need to be able to access the data of websites (their novels)
    a. This can be done through Official APIs or Unofficial Ones
        i. Most novel websites don't have official APIs so I need to use unofficial ones.
            1. Unofficial APIs aren't technically legal, but not really illegal
            2. It might be hard to get/find one current enough
            3. Utilizing/Changing it everytime might result in negative consequences
                a. I.E. the website will need to recatalogue and categorize everything, which will take tons of time and effort
        ii. I can potentially partner up with websites that do have official APIs
            1. This will allow for me to gain their data without having to use more offhanded means (data scraping & unofficial APIs)
                a. I have no idea what terms they will impose upon me
                    i. Most websites with official APIs usually have paid content and ads
                        1. I would like to be ad-free or at least not push sponsored content on to people
                        2. Allow people to have more freedom in the novels they want to explore

2. I can pivot and turn my website into a scrapping website
    a. Basically a pirate website
        i. This allows me to gain more data to use and provide for content
            1. I don't want to do this since I'll feel bad for the authors that worked hard to produce their content and post it upon a website of their own choosing
    b. Another way is to pivot in the opposite direction and allow for others to create novels on my website
        i. It will be hard to attract new people
            1. Preexisting novel websites have existed for a while now, and especially after COVID, users have already predetermined which one is their favorite to use
3. There are a lot of niche tags, more than 100, and I might not have enough content for it. Plus the description for each tag will be hard to implement. There are still more bugs that I need to account for and think/work to fix/improve.