Solar-Sentinel: A Real-Time UV Index & Sun Protection Web Application

Submitted By:

Sukhada Bhoyar

sukhadabhoyar@gmail.com

For:

Agnirva Software Internship Program - Summer 2025

Date of Submission:

June 28, 2025

Abstract

Solar-Sentinel is an innovative web application meticulously designed to empower users with critical real-time UV index information and highly personalized sun protection guidance. In an era profoundly marked by increasing environmental awareness, shifting climate patterns, and a heightened focus on personal health, understanding and effectively mitigating the inherent risks associated with ultraviolet (UV) radiation has become an imperative. This project directly addresses a significant societal need for an accessible, interactive, and comprehensive educational tool that extends far beyond the rudimentary capabilities of conventional weather forecasts. By seamlessly integrating live UV data fetched from a reliable external API, offering an exhaustive and easily digestible protection guide, providing an intuitive skin type quiz for deeply tailored advice, and implementing a crucial, user-friendly sunscreen reminder system, Solar-Sentinel aims to proactively foster responsible and consistent sun exposure habits among its users. Developed using a robust and modern web stack, comprising HTML for structure, CSS (specifically Tailwind CSS) for sleek and responsive styling, vanilla JavaScript for dynamic interactivity, and a lightweight Python Flask backend for serving content and securely managing API interactions with the OpenUV API, the application delivers a seamless, intuitive, and highly engaging user experience. Ultimately, Solar-Sentinel contributes significantly to public health awareness campaigns dedicated to promoting optimal skin health and safety under the sun.

1. Introduction

Ultraviolet (UV) radiation, an invisible component of sunlight, plays a dual role in human health. While it is undeniably essential for vital processes such as Vitamin D synthesis, which supports bone health and immune function, prolonged and unprotected exposure carries substantial and well-documented health risks. These risks encompass a spectrum of conditions, ranging from the immediate discomfort of sunburn and accelerated skin aging (manifesting as wrinkles, sunspots, and loss of elasticity) to severe long-term consequences, most notably various forms of skin cancer (melanoma, basal cell carcinoma, and squamous cell carcinoma) and irreversible eye damage (such as cataracts and pterygium). As global climate patterns continue to evolve and modern lifestyles increasingly involve outdoor activities, individuals are becoming more acutely aware of these dangers and are actively seeking reliable information and sophisticated tools to manage their sun exposure effectively. However, a pervasive limitation exists within the current landscape of digital solutions: traditional weather applications, while ubiquitous, typically offer only limited or highly generalized UV information. They often fall short in providing personalized advice, actionable protection strategies, or the interactive educational content necessary for truly informed decision-making.

Solar-Sentinel was meticulously conceived and developed to directly address this critical gap. It stands as a dedicated web application, singularly focused on comprehensive UV safety, designed from the ground up to be a user-friendly, comprehensive, and empowering resource for anyone committed to safeguarding themselves and their loved ones from the detrimental effects of harmful UV rays. The core purpose of this project is multifaceted: to provide precise, real-time, and location-specific UV data, synergistically combined with rich educational content and practical, interactive tools. This integrated approach is engineered to actively encourage and reinforce proactive sun protection behaviors, transforming passive awareness into active habit formation.

Problem Statement:

The prevailing challenge lies in the widespread lack of easily accessible, truly personalized, and actionable UV index information. This deficiency is compounded by insufficient public awareness regarding comprehensive and effective sun protection practices. The collective outcome of these factors is a demonstrable increase in the incidence and severity of UV-related health issues across populations. Existing digital solutions, while numerous, frequently suffer from a critical absence of interactive features, the provision of genuinely tailored advice, or robust, consistent reminder systems. This makes it inherently challenging for individuals to consistently adopt and maintain safe sun habits, leading to preventable health consequences and a reactive rather than proactive approach to sun exposure. Solar-Sentinel aims to provide a proactive solution by integrating these missing elements into a cohesive and engaging platform.

2. Objective

The primary objectives that guided the development of the Solar-Sentinel project are meticulously defined to ensure a comprehensive and impactful solution:

- **To develop a user-friendly web application that provides real-time UV index data for selected locations.** This objective focuses on creating an intuitive interface where users can effortlessly select their desired city (e.g., New York, Nagpur, London) and immediately receive accurate, up-to-the-minute UV index readings. The emphasis here is on simplicity and immediate utility, ensuring that critical information is readily available without unnecessary complexity.
- **To offer dynamic, personalized recommendations for sun protection (e.g., sunscreen SPF, sun avoidance times) based on the current UV index and user's skin type.** This goes beyond merely displaying a number. The application aims to translate the raw UV index into actionable advice. For instance, a high UV index would trigger recommendations for higher SPF sunscreen and suggestions to seek shade during specific hours, with these recommendations further refined by the user's unique Fitzpatrick skin type, ensuring relevance and effectiveness.
- **To educate users about the effects of UV radiation and best practices for sun safety through a comprehensive protection guide.** This objective underscores the educational mission of Solar-Sentinel. The protection guide will serve as a rich repository of information, detailing the scientific basis of UV harm, explaining various sun protection methods, and offering practical tips that empower users to understand *why* certain precautions are necessary, fostering long-term behavioral change.
- **To implement an interactive skin type quiz to help users understand their susceptibility to UV damage and receive tailored advice.** Recognizing that sun sensitivity varies greatly among individuals, this quiz is designed to be a personalized assessment tool. By answering a few simple questions, users can determine their Fitzpatrick skin type, which then allows the application to provide highly specific and relevant sun protection advice, moving away from generic recommendations.
- **To create a functional sunscreen reminder system to promote consistent reapplication habits.** One of the biggest challenges in sun protection is consistent reapplication. This objective aims to build a practical tool that allows users to set custom reminders based on their activity levels and sunscreen type, ensuring they are prompted to reapply at optimal intervals, thereby maintaining effective protection throughout the day.
- **To visualize UV forecast data to help users plan their outdoor activities**

safely. Beyond current conditions, understanding future UV levels is crucial for planning. This objective involves presenting a clear, graphical representation of the UV index forecast for the upcoming hours, enabling users to anticipate peak UV times and adjust their outdoor schedules accordingly, promoting proactive safety.

- **To ensure the application is responsive and accessible across various devices.** In today's multi-device world, an application's utility is significantly enhanced by its adaptability. This objective ensures that Solar-Sentinel provides an optimal viewing and interaction experience whether accessed on a smartphone, tablet, or desktop computer, making it universally available to a broad user base.

3. Literature Review

The scientific understanding of ultraviolet (UV) radiation's profound effects on human health has undergone significant advancements over decades. UV radiation, a segment of the electromagnetic spectrum, is broadly categorized into three primary types based on wavelength: UVA (320-400 nm), UVB (290-320 nm), and UVC (100-290 nm). While UVC radiation is almost entirely absorbed by the Earth's ozone layer and poses minimal direct threat to surface life, both UVA and UVB radiation successfully penetrate the atmosphere and reach the Earth's surface, contributing distinctly to skin damage and other health concerns. UVA rays, which constitute the majority of UV radiation reaching us, are primarily responsible for premature skin aging, wrinkles, and play a significant role in the development of skin cancer. UVB rays, though less abundant, are the primary cause of sunburn and directly contribute to DNA damage, significantly increasing the risk of skin cancer.

The **UV Index (UVI)** serves as a globally standardized measure of the strength of sun-burning UV radiation at a particular place and time. Developed collaboratively by the World Health Organization (WHO), the United Nations Environment Programme (UNEP), the World Meteorological Organization (WMO), and the International Commission on Non-Ionizing Radiation Protection (ICNIRP), the UVI provides a simple, numerical scale (typically 0 to 11+) to indicate the potential for skin damage. A higher UVI value unequivocally signifies a greater risk of skin damage and a more rapid onset of sunburn, necessitating increased protective measures. For instance, a UVI of 1-2 indicates low risk, 3-5 moderate, 6-7 high, 8-10 very high, and 11+ extreme.

Extensive epidemiological and clinical research consistently demonstrates a strong, undeniable correlation between chronic and excessive UV exposure and the escalating incidence of various forms of skin cancers, including the highly aggressive melanoma, as well as basal cell carcinoma and squamous cell carcinoma. Beyond oncological risks, UV radiation is a primary extrinsic factor accelerating skin aging, leading to characteristic signs such as fine lines, deep wrinkles, hyperpigmentation (e.g., sunspots, age spots), and a noticeable loss of skin elasticity. Furthermore, ocular conditions like cataracts (clouding of the eye's lens) and pterygium (a benign growth on the conjunctiva) are unequivocally linked to prolonged and unprotected UV exposure, highlighting the systemic impact of solar radiation. UV exposure can also suppress the immune system, making the body more vulnerable to infections and reducing the effectiveness of vaccinations.

In response to these pervasive health threats, public health campaigns globally

emphasize the critical importance of adopting comprehensive sun protection measures. These include, but are not limited to, strategically seeking shade, particularly during peak UV hours (typically 10 AM to 4 PM), wearing protective clothing (long-sleeved shirts, wide-brimmed hats, UV-blocking sunglasses), consistently applying broad-spectrum sunscreen with an adequate Sun Protection Factor (SPF) of 30 or higher, and understanding the necessity of regular reapplication. The **Fitzpatrick skin type classification system**, developed by Thomas B. Fitzpatrick in 1975, is a widely adopted dermatological tool used to categorize human skin based on its response to sun exposure, specifically its tendency to burn or tan. This six-point scale (Type I to Type VI) provides a crucial basis for dermatologists and public health professionals to offer personalized sun protection recommendations, acknowledging the inherent variations in melanin content and UV sensitivity across individuals.

While numerous applications currently exist that provide weather forecasts, many of which now include a basic UV index reading, a significant functional and informational gap persists. Many of these general-purpose apps often lack the depth of personalized advice, the richness of interactive educational content, or the robustness of a consistent reminder system that Solar-Sentinel is specifically engineered to provide. Some niche applications focus exclusively on singular aspects, such as merely providing sunscreen reapplication reminders, while others are broad-spectrum weather apps where UV information is a secondary, often underexplored, feature. Solar-Sentinel distinguishes itself by seeking to seamlessly integrate these disparate, yet crucial, functionalities into a single, comprehensive, user-centric, and highly intuitive platform, thereby offering a holistic solution to effective sun safety management.

4. System Architecture / Flow Diagram

The Solar-Sentinel application is meticulously structured around a robust **client-server architecture**, designed for efficient data flow and clear separation of concerns. At its core, a lightweight Python Flask backend serves as the central orchestrator, primarily responsible for delivering the static frontend assets to the user's browser and securely managing critical API interactions.

Client-Side (Frontend):

The frontend represents the user-facing component of the application, built with standard web technologies to ensure broad compatibility and a rich interactive experience:

- **HTML (content.html):** This foundational layer provides the semantic and structural layout of all the web pages. It meticulously defines the various sections of the application, including the dynamic dashboard, the informative sun hours display, the analytical UV forecast chart, the comprehensive protection guide, the interactive skin type quiz, and the practical reminder system. Its role is to organize content logically for presentation.
- **CSS (styles.css & Tailwind CSS):** The visual aesthetics and responsive behavior of Solar-Sentinel are primarily governed by CSS.
    - **Custom CSS (styles.css):** This file contains specific, hand-crafted styling rules, intricate animations (such as fadeIn for smooth transitions), and unique component designs (like the uv-meter and its uv-indicator) that provide the application with its distinct visual identity and polished feel.
    - **Tailwind CSS:** As a utility-first CSS framework, Tailwind CSS is extensively utilized for rapid UI development. It provides a vast array of pre-defined, atomic classes for managing layout, spacing, typography, color palettes, and crucially, responsive design breakpoints. This approach ensures a consistent, modern, and adaptive user interface that gracefully adjusts across a multitude of devices, from small mobile screens to large desktop monitors.
- **JavaScript (script.js):** This is the dynamic brain of the client-side, handling all interactive elements and application logic. Its responsibilities are extensive and critical:
    - **Data Fetching:** Initiates and manages asynchronous requests to fetch real-time UV data from the OpenUV API (or potentially from the Flask backend acting as a proxy).
    - **DOM Manipulation:** Dynamically updates the HTML elements in response to user interactions (e.g., city selection from a dropdown, submission of quiz answers) and incoming data, ensuring the UI is always current and reflective

of the application's state.
    - **Real-Time Clock:** Implements and continuously updates a precise real-time clock, providing users with accurate temporal context.
    - **Tab Navigation:** Manages the seamless switching between different content tabs (Dashboard, Sun Hours, etc.), ensuring a smooth user flow without full page reloads.
    - **Chart Rendering:** Utilizes the Chart.js library to draw, update, and manage the interactive bar graph that visualizes the UV forecast data.
    - **Quiz Logic:** Contains the algorithms for processing quiz answers, determining the user's Fitzpatrick skin type, and subsequently displaying personalized recommendations.
    - **Sunscreen Reminder Functionality:** Handles the setting, scheduling, and management of user-defined sunscreen reapplication reminders, including triggering alerts.

Server-Side (Backend):

The backend provides the necessary server infrastructure and security for the application:

- **Python (app.py):** This Flask application serves as the core server component.
    - **File Serving:** Its primary role is to serve the index.html template, which in turn loads all the necessary frontend assets (CSS, JavaScript).
    - **API Key Management:** Crucially, it manages the OPENUV_API_KEY securely. By loading this sensitive key from environment variables (using python-dotenv), it prevents direct exposure of the key in the client-side code, enhancing security. While the current script.js directly calls OpenUV, a more robust production-grade setup would involve the Flask backend acting as a secure proxy for all external API calls. This would further abstract the API key from the client and allow for server-side rate limiting or data caching if needed.

API Integration:

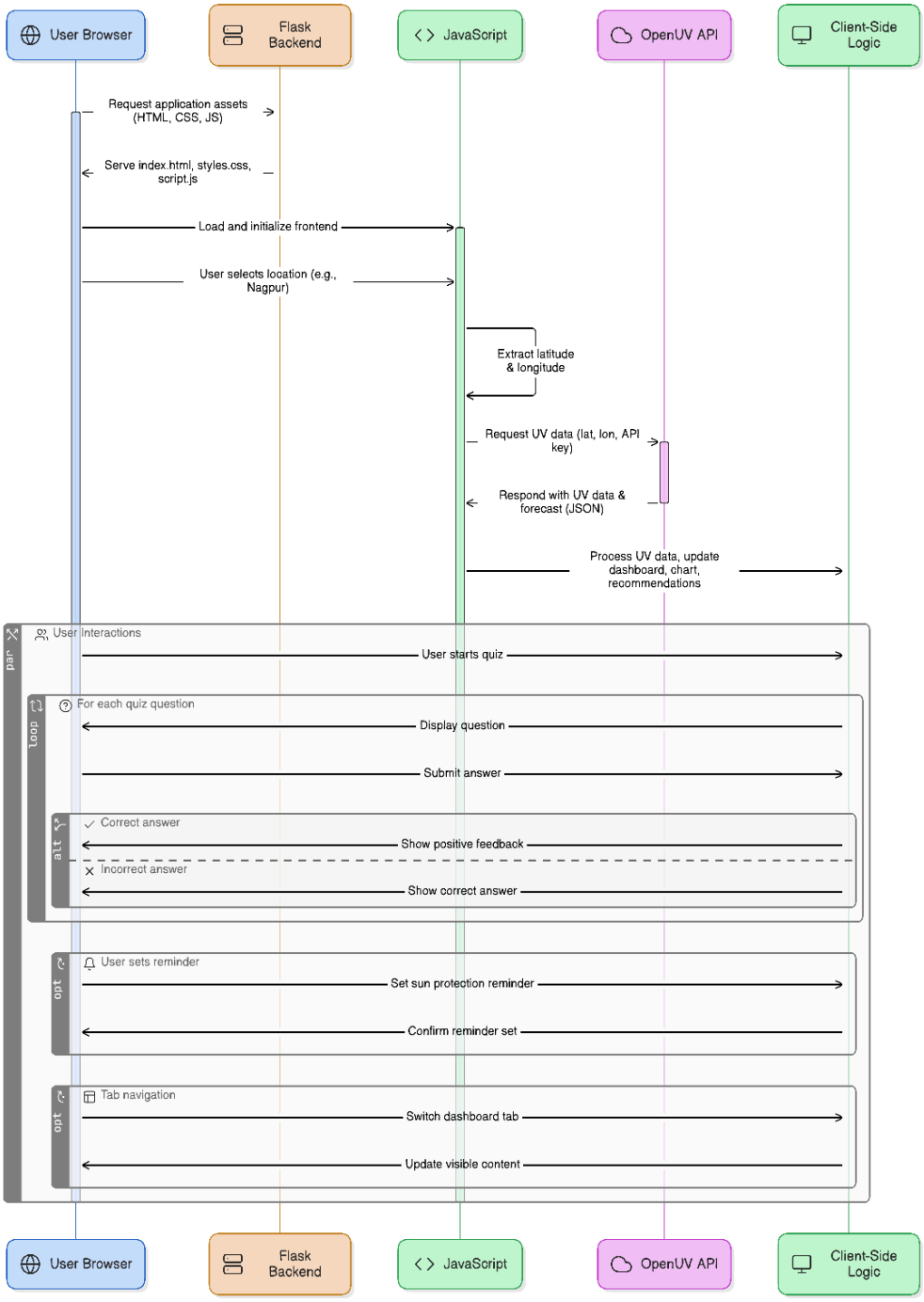
- **OpenUV API:** This is the indispensable third-party external API that forms the backbone of the application's data. It provides up-to-the-minute real-time UV index data and comprehensive forecast information for any specified geographical coordinates (latitude and longitude). The reliability and accuracy of this API are paramount to Solar-Sentinel's core functionality.

Data Flow:

The interaction and information exchange within Solar-Sentinel follow a clear, sequential data flow:

1. **User Request:** The journey begins when a user initiates access to the Solar-Sentinel web application by navigating to its URL in their web browser.
2. **Frontend Load:** The Flask backend (app.py) receives this initial request and responds by serving the index.html template. This HTML file, in turn, references and loads all the necessary client-side assets: content.html (for structure), styles.css (for styling), and script.js (for interactivity).
3. **Location Selection:** Once the frontend is fully loaded and rendered in the user's browser, the user actively interacts with the application by selecting a specific city from a provided dropdown menu.
4. **API Call (Frontend):** Upon city selection, the JavaScript code within script.js dynamically constructs an API request. This request is directed to the OpenUV API endpoint, incorporating the geographical coordinates (latitude and longitude) corresponding to the selected city, along with the securely provided API key.
5. **Data Retrieval:** The OpenUV API receives the request, processes it, and responds by transmitting the current UV index data and relevant forecast information back to the client-side JavaScript.
6. **Data Processing & Display (Frontend):** The JavaScript code then meticulously processes the received UV data.
    - The current UV index value and its associated risk level (e.g., "Moderate," "High") are prominently displayed on the main dashboard, providing immediate insights.
    - The parsed UV forecast data is then fed into the Chart.js library, which dynamically updates the bar graph, offering a visual projection of future UV levels.
    - Crucially, personalized sun protection recommendations (e.g., suggested SPF, peak sun avoidance times) are dynamically generated and updated on the UI, adapting instantly to the current UV index.
7. **User Interaction:** Beyond the core UV data display, users can further interact with other specialized features of the application, such as the Protection Guide, the Skin Type Quiz, and the Sunscreen Reminder system. All these interactions and their underlying logic are handled efficiently and responsively on the client-side by the JavaScript, ensuring a fluid and engaging user experience without requiring constant server round-trips.

Simplified Flow Diagram:



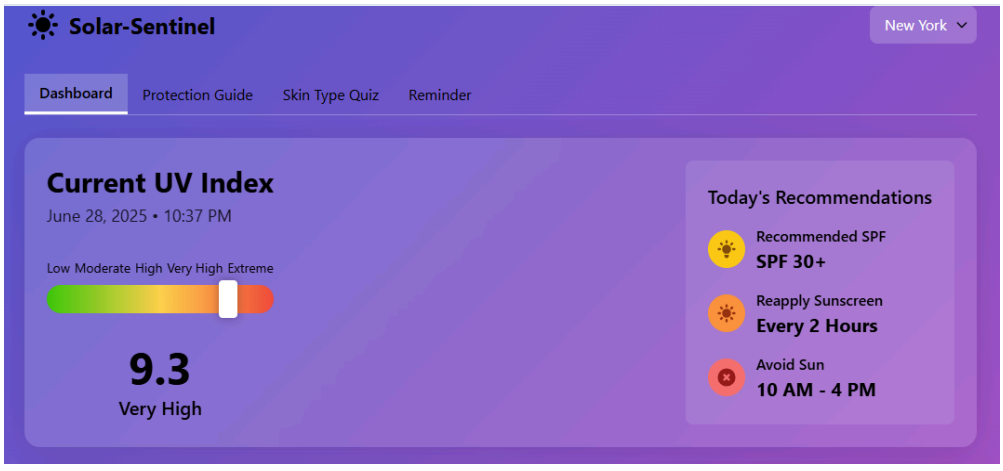
This architecture ensures a clear separation of frontend presentation from backend logic, promoting maintainability, scalability, and enhanced security by centralizing API key management.

5. Modules & Features Description

Solar-Sentinel is thoughtfully organized into several distinct and interconnected modules, each meticulously designed to provide specific functionalities that contribute to a holistic sun safety experience:

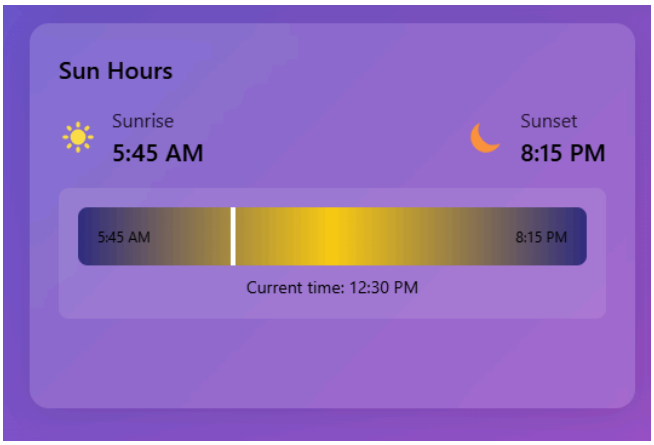
5.1. UV Dashboard

- **Description:** The UV Dashboard serves as the application's central command center and initial landing page. It is engineered to provide users with immediate, critical UV information at a glance, acting as the primary interface for understanding current sun conditions. Its design prioritizes clarity and instant comprehension.
- **Functionality:**
    - **City Selection:** Features a prominent dropdown menu allowing users to easily select their desired city from a predefined list (e.g., New York, Nagpur, Mumbai, Delhi, London). This ensures that the displayed UV data is geographically relevant to the user's current or intended location.
    - **Current UV Index & Risk Level:** Displays the real-time UV Index value numerically, accompanied by a clear, color-coded risk level indicator (e.g., "Low," "Moderate," "High," "Very High," "Extreme"). This visual cue instantly communicates the severity of the current UV radiation.
    - **Real-Time Clock:** Integrates a precise, continuously updating real-time clock showing the current date and time. This helps users contextualize the UV readings within their immediate temporal environment.
    - **Dynamic Recommendations:** A core intelligent feature of the dashboard. Based on the current UV Index, the application dynamically generates and displays personalized recommendations for appropriate sunscreen SPF levels and advice on optimal sun avoidance times. For instance, a "High" UV Index might suggest "SPF 50+ and seek shade between 10 AM - 4 PM," adapting as the UV level changes throughout the day. This proactive guidance empowers users to make informed decisions without needing to interpret raw data.
- Screenshot:



5.2. Sun Hours Section

- **Description:** This module offers a visually intuitive representation of the sun's trajectory and presence throughout the day. It helps users understand the duration of daylight and how the current time fits within it, providing a temporal context for sun exposure.
- **Functionality:**
  - **Estimated Sunrise and Sunset Times:** Displays the approximate times for sunrise and sunset for the selected location. While currently static (a known challenge), the intent is to provide accurate, location-specific timings.
  - **Progress Bar:** A highly visual progress bar graphically indicates the current time's position within the sunlit hours. As the day progresses, the bar fills up, offering an at-a-glance understanding of how much daylight remains or how much has passed. This visual aid is particularly useful for planning outdoor activities.
- Screenshot:

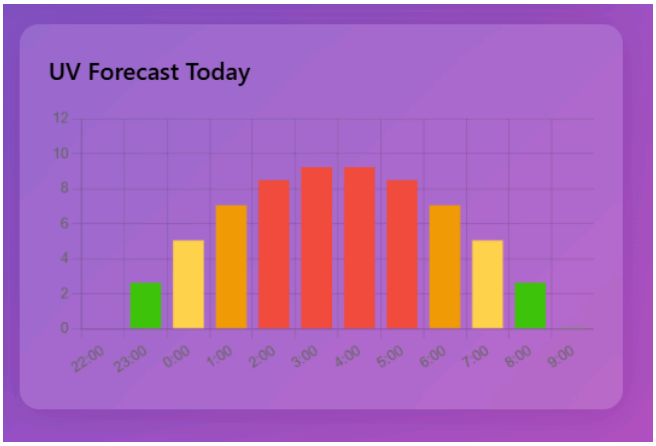


5.3. UV Forecast Chart

- **Description:** The UV Forecast Chart module transforms complex UV data into an easily digestible graphical format, allowing users to anticipate future UV levels and plan accordingly.
- **Functionality:**
  - **Chart.js Integration:** Leverages the powerful Chart.js library to display a clear bar graph. Each bar represents a simulated UV index value for the upcoming 12 hours, providing a forward-looking perspective.
  - **Color-Coded Bars:** The bars are intelligently color-coded based on the UV risk categories (e.g., green for low, yellow for moderate, orange for high, red for very high/extreme). This visual distinction allows for rapid assessment of future risk.
  - **Dynamic Updates:** The chart updates dynamically whenever the user changes the selected location, ensuring that the forecast always reflects the

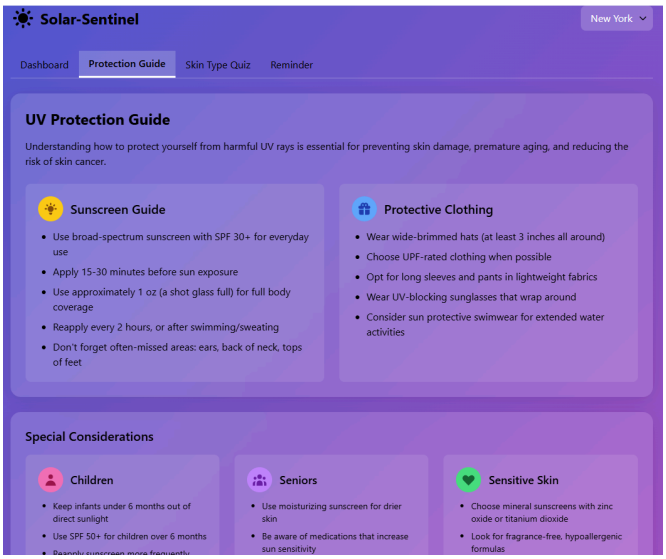
chosen city's conditions. This feature is crucial for users who might be planning travel or activities in different areas.

- Screenshot:



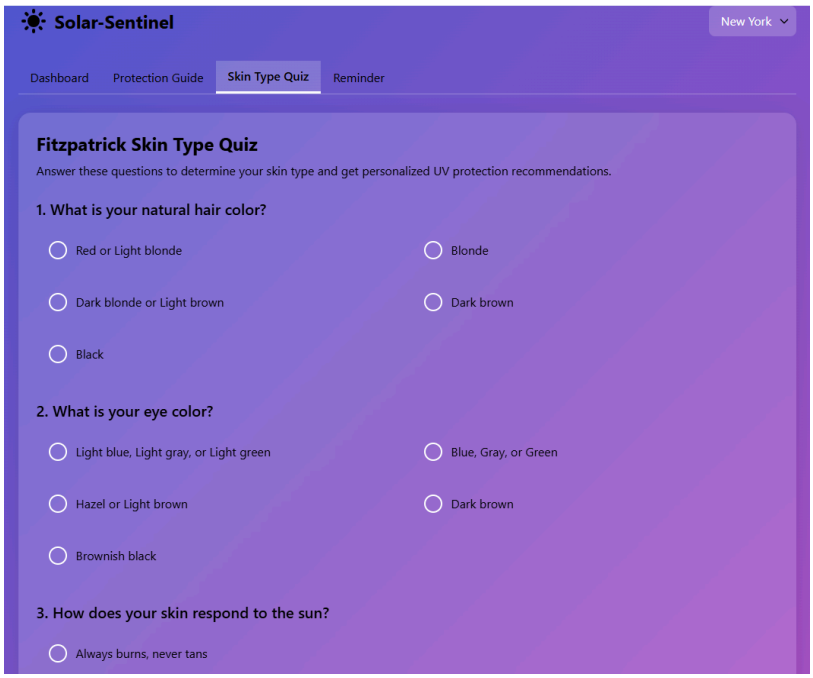
5.4. Protection Guide

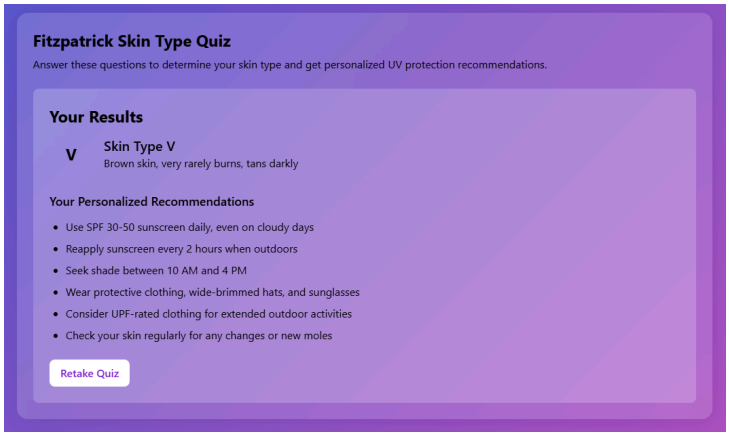
- **Description:** This module serves as an invaluable educational resource, offering comprehensive and actionable advice on best practices for sun safety and effective sun protection. It aims to empower users with knowledge, not just data.
- **Functionality:**
  - **Detailed Sunscreen Tips:** Provides in-depth guidance on proper sunscreen application techniques, including how much to use, areas often missed, and the importance of broad-spectrum protection and water resistance. It also emphasizes the critical need for reapplication.
  - **General Sun Safety:** Offers advice on complementary protection methods such as seeking shade, wearing protective clothing (e.g., UPF-rated apparel, wide-brimmed hats, UV-blocking sunglasses), and strategically avoiding outdoor activities during peak UV hours.
  - **Tailored Advice:** Recognizes that sun protection needs vary. This section includes specific recommendations for vulnerable demographics, such as children (e.g., infants under 6 months should avoid direct sun), seniors (due to thinner skin), and individuals with particularly sensitive skin types or certain medical conditions.
- Screenshot:



5.5. Skin Type Quiz

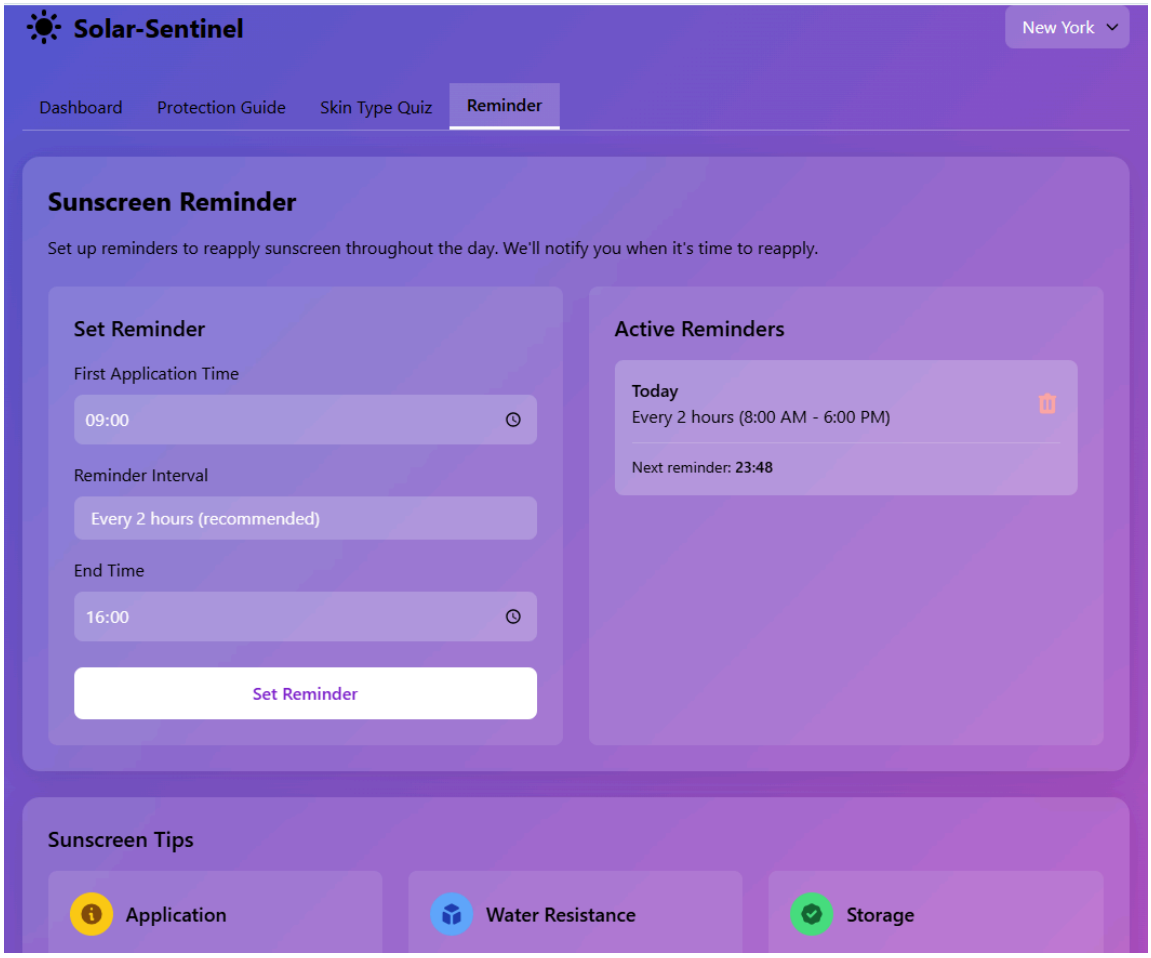
- **Description:** An interactive and engaging quiz designed to help users understand their personal susceptibility to UV damage by identifying their Fitzpatrick skin type. This personalization is key to providing truly relevant advice.
- **Functionality:**
  - **Three-Question Format:** A concise and easy-to-follow multiple-choice quiz consisting of three questions, designed to quickly assess an individual's typical reaction to sun exposure (e.g., how easily they burn or tan).
  - **Fitzpatrick Skin Type Calculation:** Based on the user's answers, the quiz algorithm calculates and displays their corresponding Fitzpatrick skin type (ranging from Type I, which always burns and never tans, to Type VI, which rarely burns and tans darkly).
  - **Personalized Recommendations:** Crucially, upon determining the skin type, the application provides highly personalized sun protection recommendations. This might include specific SPF suggestions, advice on initial sun exposure times, and warnings about particular risks relevant to their skin's melanin levels.
- Screenshot:





5.6. Sunscreen Reminder System

- **Description:** A practical and essential tool aimed at addressing one of the most common challenges in sun protection: remembering to reapply sunscreen consistently throughout the day.
- **Functionality:**
  - **Customizable Reminders:** Allows users to set a specific start time for their sun exposure, define a reapplication interval (e.g., every 2 hours), and specify an end time for the reminders. This flexibility accommodates various schedules and activities.
  - **Visual Schedule Display:** Once set, the system visually displays the scheduled reminder times, providing a clear overview of when reapplication is due. This helps users plan their day and ensures they maintain continuous protection, even during extended outdoor activities.
- Screenshot:



6. Technologies Used

The Solar-Sentinel application is built upon a carefully selected stack of modern web technologies, chosen for their efficiency, flexibility, and robust capabilities. This combination ensures a responsive, interactive, and maintainable application.

Frontend

The frontend is responsible for everything the user sees and interacts with in their web browser.

- **HTML (HyperText Markup Language):** This is the foundational language for creating web pages. In Solar-Sentinel, HTML (content.html) provides the semantic structure of all content, defining elements like headings, paragraphs, lists, forms, and containers for dynamic content. It ensures the logical organization of information, making the application accessible and readable by browsers and assistive technologies.
- **CSS (Cascading Style Sheets) - Custom & Tailwind CSS:** CSS is vital for the visual presentation and aesthetic appeal of the application.
    - **Custom CSS (styles.css):** This file contains specific, hand-written CSS rules for unique design elements, custom animations (e.g., fadeIn for smooth transitions between tabs), and bespoke component styling (like the visually distinctive uv-meter and its uv-indicator). It allows for fine-grained control over the application's unique look and feel.
    - **Tailwind CSS:** A utility-first CSS framework, Tailwind CSS accelerates UI development by providing a vast collection of low-level utility classes directly within the HTML. Instead of writing custom CSS for every element, developers can compose designs by combining classes like flex, p-4, text-lg, rounded-lg, and responsive prefixes (md:flex). This approach promotes consistency, reduces CSS bloat, and significantly speeds up the styling process, while ensuring the application is inherently responsive and adapts gracefully to various screen sizes.
- **JavaScript (Vanilla):** As the core scripting language for the client-side, vanilla JavaScript (meaning no large frameworks like React or Angular) powers all the dynamic behaviors and interactive elements of Solar-Sentinel. It handles:
    - **DOM Manipulation:** Dynamically adding, removing, and modifying HTML elements based on user input or data updates.
    - **Event Handling:** Responding to user actions such as clicks, selections, and form submissions.

    - **Asynchronous Operations:** Making API calls to fetch data without blocking the user interface.
    - **Client-Side Logic:** Implementing the quiz logic, managing the real-time clock, and scheduling sunscreen reminders. Its choice ensures a lightweight and performant frontend.
- **Chart.js:** This is a powerful and flexible open-source JavaScript charting library. It's specifically integrated into Solar-Sentinel to create the interactive and visually appealing bar chart that displays the UV forecast. Chart.js simplifies the process of rendering complex data into clear, understandable graphical formats, enhancing the user's ability to interpret and utilize the forecast information.

Backend

The backend handles server-side operations, including serving the web application and managing sensitive information.

- **Python:** A versatile, high-level, interpreted programming language known for its readability and extensive libraries. Python is chosen for the server-side logic due to its simplicity, robust ecosystem, and suitability for web development.
- **Flask:** A lightweight micro web framework for Python. Flask is preferred for Solar-Sentinel due to its minimalist design, which allows developers to build web applications quickly without unnecessary overhead. It's ideal for serving static files (like HTML, CSS, JS) and handling basic routing and API key management, making it a perfect fit for a focused application like Solar-Sentinel. Its "micro" nature means it doesn't enforce specific tools or libraries, offering great flexibility.

API

- **OpenUV API:** This is a crucial third-party external API that provides the essential real-time and forecast UV index data. The API allows fetching UV information for any given geographical coordinates (latitude and longitude). Its reliability and comprehensive data endpoints are fundamental to the core functionality of Solar-Sentinel, enabling the application to provide accurate and up-to-date UV insights.

Hosting

- **Render.com:** A modern cloud platform that simplifies the deployment and scaling of web applications, databases, and services. Render.com is used to host the Flask application, providing a live and accessible URL for the Solar-Sentinel web application, making it available to users globally without complex infrastructure management.

- **Website:** https://solar-sentinel.onrender.com/

Miscellaneous

- **python-dotenv:** A Python library designed to read key-value pairs from a .env file and set them as environment variables. This is a critical security measure in Solar-Sentinel, ensuring that sensitive information like the OPENUV_API_KEY is not hardcoded directly into the source code but is instead loaded securely at runtime, preventing its exposure in public repositories.
- **Dynamic DOM Manipulation:** A core JavaScript technique extensively employed throughout the application. It allows for the modification of the web page's structure, style, and content in response to user actions or data changes, creating a highly interactive and responsive user experience without full page reloads.
- **Local Time Management:** JavaScript functions are implemented to accurately display the current date and time, and to manage time-based events such as the scheduling and triggering of sunscreen reminders, ensuring they are delivered at the appropriate moments.
- **Quiz Logic:** The JavaScript code incorporates the logic necessary for evaluating user answers in the Skin Type Quiz, calculating the corresponding Fitzpatrick skin type, and subsequently delivering personalized sun protection recommendations, making the quiz an intelligent and valuable feature.

7. Challenges Faced & Solutions

During the development of Solar-Sentinel, several challenges were encountered, ranging from specific functional limitations to broader architectural considerations. Addressing these challenges, or proposing clear solutions for future implementation, was crucial for the project's integrity and potential for growth.

7.1. Challenge: Static Sun Hours Card

- **Description:** The "Sun Hours" card, intended to provide a dynamic visual representation of daylight hours, currently displays static sunrise and sunset times. This means the times shown do not update in real-time based on the user's selected location, nor do they reflect the actual astronomical sunrise/sunset for the current date. This significantly limits the utility and accuracy of this particular feature, as users cannot rely on it for precise planning.
- **Implications:** Without real-time data, the sun hours display becomes a mere placeholder, potentially misleading users about actual daylight periods, especially when changing locations or viewing on different days. This diminishes the application's perceived accuracy and value.
- **Solution (Proposed):** To rectify this, the application would require integration with a dedicated external API that provides accurate sunrise and sunset times for any given latitude and longitude. APIs such as the Sunrise-Sunset API (e.g., api.sunrise-sunset.org) or potentially more comprehensive weather APIs that include astronomical data could be utilized. The JavaScript code in script.js would then be modified to:
  1. Make an asynchronous API call to this new endpoint whenever a new city is selected.
  2. Pass the latitude and longitude of the selected city, along with the current date, to the API.
  3. Parse the returned sunrise and sunset times.
  4. Dynamically update the text content displaying these times on the "Sun Hours" card.
  5. Adjust the visual progress bar to accurately reflect the current time's position within the actual sunlit period, providing a truly dynamic and informative experience.

7.2. Challenge: Non-Functional Reminder Delete Button

- **Description:** A critical usability flaw exists within the "Sunscreen Reminder" section: the delete button associated with individual reminders is currently non-functional. Users are unable to remove previously set reminders from the

displayed list, nor does clicking the button stop any associated background alerts or timers. This leads to a cluttered interface and a lack of control for the user.

- **Implications:** The inability to delete reminders creates a frustrating user experience. It clogs the reminder list with obsolete entries, and more importantly, it means that scheduled alerts might continue to fire even if the user no longer needs them, leading to annoyance and potentially causing users to abandon the feature entirely.
- **Solution (Proposed):** The JavaScript code within script.js needs a robust implementation for reminder deletion. This would involve:
    1. **Event Listener:** Attaching an event listener (e.g., click event) to each dynamically created delete button.
    2. **Reminder Identification:** When a delete button is clicked, the JavaScript must accurately identify which specific reminder is targeted for deletion. This can be achieved by assigning a unique ID to each reminder when it's created and associating this ID with its corresponding delete button (e.g., via a data-id attribute).
    3. **Data Structure Removal:** The identified reminder object must be removed from the internal data structure where reminders are stored (e.g., an array of reminder objects).
    4. **DOM Update:** The corresponding HTML element representing the deleted reminder must be removed from the Document Object Model (DOM) to visually update the list.
    5. **Timer Clearance:** Crucially, any active JavaScript setInterval or setTimeout calls associated with that specific reminder must be cleared using clearInterval() or clearTimeout() to prevent it from triggering future alerts. This ensures a clean and complete deletion process.

7.3. Challenge: API Key Exposure (Potential)

- **Description:** While the Flask backend loads the OPENUV_API_KEY from environment variables using python-dotenv, the script.js directly utilizes this api_key which is rendered into the index.html template. This means that the API key, although loaded securely on the server, is ultimately exposed in the client-side source code (visible in the browser's developer tools). This is a significant security vulnerability, as malicious actors could extract and misuse the key.
- **Implications:** Exposure of the API key could lead to unauthorized usage of the OpenUV API, potentially incurring unexpected costs or hitting rate limits for the legitimate application. It compromises the security posture of the application and the integrity of the API usage.

- **Solution (Proposed):** The most effective solution is to implement a **backend proxy**. Instead of the frontend directly making calls to the OpenUV API, the frontend would make requests to a new, secure endpoint on the Flask backend. The Flask backend would then:
  1. Receive the request from the frontend (e.g., for UV data for a specific latitude/longitude).
  2. Securely make the actual API call to the OpenUV API using its internally stored OPENUV_API_KEY (which is never exposed to the client).
  3. Receive the response from OpenUV.
  4. Process the response (if necessary) and return the relevant data to the frontend.
     This architecture ensures the API key remains server-side, significantly enhancing security.

7.4. Challenge: Simulated UV Forecast

- **Description:** The current UV forecast displayed in the chart is simulated using a simple sinusoidal mathematical model. This means it does not reflect actual, real-world predicted UV levels from the OpenUV API, but rather a generic pattern. While useful for demonstration, it lacks accuracy and practical utility for real-world planning.
- **Implications:** A simulated forecast, while visually appealing, provides inaccurate information. Users relying on it for planning outdoor activities might be exposed to higher or lower UV levels than anticipated, leading to ineffective sun protection or unnecessary caution. This undermines the application's core value proposition of providing reliable UV data.
- **Solution (Proposed):** The script.js needs to be modified to parse and utilize the actual forecast data provided by the OpenUV API. The OpenUV API offers a uv_forecast endpoint (or similar) that, when queried with latitude, longitude, and a future date/time range, returns predicted UV index values. The steps would involve:
  1. Making an API call to the OpenUV forecast endpoint.
  2. Parsing the JSON response, which contains hourly or daily UV forecast data.
  3. Extracting the uv_time and uv values from the forecast array.
  4. Feeding these real forecast labels and values into the updateChart() function, replacing the simulateForecast() logic. This would provide users with accurate, data-driven UV predictions, significantly enhancing the application's practical utility.

8. Future Scope

The Solar-Sentinel project, while functional and impactful in its current state, possesses substantial potential for future expansion and enhancement. These proposed future scopes aim to further enrich the user experience, improve data accuracy, and broaden the application's utility, transforming it into an even more indispensable tool for sun safety.

- **Real-time Sunrise/Sunset Data Integration:** Currently, the "Sun Hours" section provides static times. A crucial enhancement would be to integrate with a dedicated astronomical API (e.g., a specific sunrise/sunset API or a more comprehensive weather API that includes this data). This would allow the application to fetch and display accurate, real-time sunrise and sunset times based on the user's selected location, making the "Sun Hours" section truly dynamic and geographically precise. This would significantly improve the utility of the visual progress bar, allowing users to accurately gauge daylight periods for planning.
- **Persistent Reminders (Local Storage / Backend Database):** The current reminder system is ephemeral, meaning reminders are lost when the browser tab is closed. Implementing local storage (for client-side persistence) or, ideally, a simple backend database (like Firestore, if user accounts are introduced) would allow sunscreen reminders to persist across browser sessions. This ensures users don't have to re-enter their reminder preferences each time they visit the app, greatly enhancing convenience and reliability.
- **User Accounts & Personalization:** Introducing user authentication (e.g., via Firebase Authentication) would unlock a new level of personalization. Users could create accounts to securely store their personalized settings, past quiz results, preferred locations, and reminder preferences. This would allow for a highly tailored experience, where the app "remembers" the user and provides immediate, relevant information upon login, fostering a stronger sense of ownership and utility.
- **Automated Location Services:** Integrating the browser's Geolocation API would enable the application to automatically detect the user's current location (with user permission). This would eliminate the need for manual city selection, providing immediate and highly relevant UV data and recommendations as soon as the app loads, significantly improving the initial user experience and ease of use.
- **Advanced UV Forecast (Multi-Day):** While the current forecast is simulated hourly, utilizing the full capabilities of the OpenUV API (or a complementary

weather API) would allow for the display of a multi-day UV forecast. This would provide users with a broader outlook on UV levels, enabling them to plan outdoor activities, vacations, or extended trips with greater foresight and confidence regarding sun safety.

- **Native Notification System:** For the sunscreen reminder system to be truly effective, implementing browser notifications (Web Notifications API) would be a significant upgrade. This would allow reminders to pop up as system-level notifications, even when the application is not actively in focus or the browser tab is minimized, ensuring that users receive timely prompts regardless of their current activity.
- **Expanded Protection Guide with Multimedia:** The protection guide could be significantly enhanced by adding more in-depth articles, engaging infographics, short instructional videos, and interactive content. Topics could expand to include detailed explanations of different types of sunscreens (mineral vs. chemical), common myths about sun exposure, the science behind UV filters, and practical demonstrations of proper application techniques.
- **Integration with Wearables/Smart Devices:** Exploring potential integrations with smartwatches or fitness trackers could provide on-the-go UV alerts and reminders directly to the user's wrist. This ubiquitous access would make sun protection even more seamless and integrated into daily life, especially for active individuals.
- **Multi-language Support:** To make Solar-Sentinel accessible to a global audience, implementing multi-language support (internationalization, i18n) would be crucial. This would involve translating all UI text and content into various languages, catering to a diverse user base.
- **Accessibility Enhancements (WCAG Compliance):** Further improving accessibility features for users with disabilities (e.g., screen reader compatibility, keyboard navigation, sufficient color contrast) would ensure the application is usable by everyone, adhering to Web Content Accessibility Guidelines (WCAG).
- **Historical UV Data Tracking:** Allowing users to view historical UV data for their selected locations could provide insights into typical UV patterns, helping them understand seasonal variations and long-term exposure trends.
- **Community Features/Sharing:** Implementing features that allow users to share their UV readings, protection tips, pictures of sunset to rate the best picture of the day, or even challenge friends to maintain sun-safe habits could foster a community around sun safety, encouraging collective responsibility and engagement.

9. Conclusion

Solar-Sentinel, in its current iteration, successfully delivers on its foundational mission: to provide a user-friendly, informative, and actionable platform dedicated to promoting comprehensive sun safety. By seamlessly integrating real-time UV data with thoughtfully curated educational content, personalized advice derived from a skin type quiz, and practical tools like the crucial sunscreen reminder system, the application effectively empowers users to make informed and proactive decisions regarding their sun exposure. This proactive approach is vital in mitigating the long-term health risks associated with harmful UV radiation. While the project acknowledges and openly addresses a few identified challenges, such as the static nature of the sun hours display and the non-functional reminder deletion, these are not insurmountable obstacles. Rather, they represent clear, well-defined opportunities for future development and refinement. These planned enhancements promise to significantly bolster the application's accuracy, expand its functionality, and further elevate the overall user experience. Ultimately, Solar-Sentinel stands as a compelling testament to the transformative power of combining modern web technologies with essential public health initiatives, fostering healthier habits and contributing meaningfully to the global effort to mitigate environmental health risks. Its intuitive design and focused utility position it as a valuable asset in the ongoing endeavor to promote responsible sun protection.

10. References

- **World Health Organization (WHO) - UV Index:** Provides comprehensive information on UV radiation, its health effects, and the global UV Index.
  - URL: https://www.who.int/news-room/fact-sheets/detail/uv-radiation-and-health
- **OpenUV API Documentation:** The official documentation for the OpenUV API, detailing endpoints for current UV index, forecast, and associated data.
  - URL: https://www.openuv.io/
- **Chart.js Documentation:** The official documentation for the Chart.js library, outlining its usage, chart types, and customization options.
  - URL: https://www.chartjs.org/docs/latest/
- **Tailwind CSS Documentation:** The comprehensive documentation for the Tailwind CSS utility-first framework, explaining its classes, configuration, and responsive design principles.
  - URL: https://tailwindcss.com/docs
- **Flask Documentation:** The official documentation for the Flask web framework, covering its core concepts, routing, templating, and deployment.
  - URL: https://flask.palletsprojects.com/
- **Fitzpatrick Skin Type Scale:** (It is recommended to cite a reputable medical or dermatological source for this, such as: American Academy of Dermatology, National Cancer Institute, or a peer-reviewed dermatology journal article.)
  - Example Source Type: *Fitzpatrick, T. B. (1988). The validity and practicality of sun-reactive skin types I through VI. Archives of Dermatology, 124(6), 869-871.* (Please replace with an actual, accessible reference if available).
- **Python-dotenv Documentation:** Documentation for the python-dotenv library, detailing how to load environment variables from .env files.
  - URL: https://pypi.org/project/python-dotenv/
- **Deployment:** The live application is deployed and accessible online at solar-sentinel.onrender.com (You're welcome to try it out anytime).

11. Appendix

Appendix A: Code Snippets

A.1. app.py (Flask Backend)

This Python script sets up a basic Flask web server. It loads environment variables, specifically the OPENUV_API_KEY, and renders an index.html template, passing the API key to the frontend. This ensures the API key is managed securely on the server side.

```
# app.py
from flask import Flask, render_template
from dotenv import load_dotenv
import os
# Load environment variables from a .env file
# This is crucial for securely managing sensitive information like API keys.
load_dotenv()
# Initialize the Flask application
app = Flask(__name__)
# Define the route for the homepage
@app.route('/')
def index():
    # Retrieve the OpenUV API key from environment variables.
    # This key is then passed to the HTML template, making it accessible on the client-side.
    api_key = os.getenv("OPENUV_API_KEY")
    # Render the 'index.html' template, which serves as the main entry point for the frontend.
    return render_template("index.html", api_key=api_key)
# Entry point for running the Flask application
if __name__ == "__main__":
    # Run the Flask app in debug mode.
    # In a production environment, debug=False should be used, and a production-ready WSGI
server
    # like Gunicorn or uWSGI should be used.
    app.run(debug=True)
```

A.2. script.js (Core JavaScript Logic - Excerpt for UV Chart)

This JavaScript excerpt demonstrates how the UV forecast chart is updated using Chart.js. It includes a function to update the chart with new data and a simulation function (which would ideally be replaced by actual API data in a future iteration) to generate sample forecast values. The backgroundColor logic dynamically colors the

bars based on UV risk levels.

```
// script.js (Excerpt for UV Chart update)
// Global variable to hold the Chart.js instance, allowing it to be destroyed and re-rendered.
let uvChart;
/**
 * Updates or creates the UV Index forecast bar chart.
 * @param {Array<Object>} forecast - An array of forecast objects, each with 'uv_time' and 'uv'
properties.
 */
function updateChart(forecast) {
    // Extract labels (hourly times) from the forecast data.
    const labels = forecast.map((f, i) => {
        const date = new Date(f.uv_time);
        // Format time as "HH:00" for chart labels.
        return `${date.getHours()}:00`;
    });
    // Extract UV index values from the forecast data.
    const values = forecast.map(f => f.uv);
    // Get the 2D rendering context of the canvas element.
    const ctx = document.getElementById("uvChart").getContext("2d");
    // If a previous chart instance exists, destroy it to prevent memory leaks and rendering
issues.
    if (uvChart) uvChart.destroy();
    // Create a new Chart.js bar chart instance.
    uvChart = new Chart(ctx, {
        type: "bar", // Specify chart type as 'bar'.
        data: {
            labels, // Set the x-axis labels (times).
            datasets: [{
                label: "UV Index Forecast", // Label for the dataset.
                data: values, // The UV index values for the bars.
                // Dynamically set background color based on UV index value for visual risk
indication.
                backgroundColor: values.map(val =>
                    val < 3 ? "#3EC70B" : // Low risk (Green)
                    val < 6 ? "#FFD24C" : // Moderate risk (Yellow)
                    val < 8 ? "#F29F05" : // High risk (Orange)
                    "#F24C3D"             // Very High/Extreme risk (Red)
                )
            }]
        },
        options: {
```

```
            scales: {
                y: {
                    beginAtZero: true, // Start y-axis from zero.
                    max: 12            // Set maximum y-axis value, as UV Index typically goes up to 11+.
                }
            },
            plugins: {
                legend: { display: false } // Hide the legend for a cleaner, more minimalist chart
appearance.
            },
            responsive: true, // Make the chart responsive to container size changes.
            maintainAspectRatio: false // Allow chart to change aspect ratio.
        }
    });
}
/**
 * Simulates a UV Index forecast using a sinusoidal model.
 * This function is a placeholder and would be replaced by actual API data in a production
environment.
 * @param {number} currentUV - The current UV index value.
 * @param {number} maxUV - The maximum UV index expected in the simulation.
 * @returns {Array<Object>} An array of simulated forecast objects.
 */
function simulateForecast(currentUV, maxUV) {
    const forecast = [];
    const now = new Date();
    // Loop to simulate forecast for the next 12 hours.
    for (let i = 0; i < 12; i++) {
        // Simple sinusoidal model to simulate UV fluctuation throughout the day.
        // Math.max(0, ...) ensures UV index doesn't go below zero.
        const uvSim = Math.max(0, maxUV * Math.sin(Math.PI * (i / 12) + Math.PI / 4));
        // Push a forecast object with time (ISO string) and simulated UV value.
        forecast.push({
            uv_time: new Date(now.getTime() + i * 3600000).toISOString(), // Add hours to current
time
            uv: parseFloat(uvSim.toFixed(1)) // Round UV value to one decimal place
        });
    }
    return forecast;
}
```

A.3. styles.css (Excerpt for Card and UV Meter Styling)

This CSS excerpt defines the styling for the general card components, providing a modern, frosted-glass effect with subtle shadows. It also styles the uv-meter and its uv-indicator, creating a visually appealing and informative display for the UV index level.

```
/* styles.css (Excerpt for card and UV meter) */
/* General styling for card components throughout the application */
.card {
    background: rgba(255, 255, 255, 0.15); /* Semi-transparent white background for a frosted glass effect */
    backdrop-filter: blur(10px); /* Applies a blur effect to the background behind the element */
    border-radius: 16px; /* Rounded corners for a softer, modern look */
    box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1); /* Subtle shadow for depth and separation from the background */
    border: 1px solid rgba(255, 255, 255, 0.18); /* A slight border for definition */
}
/* Styling for the UV meter progress bar */
.uv-meter {
    height: 30px; /* Fixed height for the meter */
    border-radius: 15px; /* Half of height to create fully rounded ends */
    /* Linear gradient background to visually represent UV intensity levels from low to high */
    background: linear-gradient(to right, #3EC70B, #FFD24C, #F24C3D);
    position: relative; /* Required for positioning the indicator child element */
    overflow: hidden; /* Ensures the indicator stays within the bounds if it were to extend */
}
/* Styling for the movable indicator on the UV meter */
.uv-indicator {
    width: 20px; /* Width of the indicator */
    height: 40px; /* Height of the indicator, slightly taller than the meter */
    background-color: white; /* White background for visibility */
    border-radius: 4px; /* Slightly rounded corners for the indicator */
    position: absolute; /* Allows precise positioning within the parent .uv-meter */
    top: -5px; /* Positions the indicator slightly above the meter's top edge */
    transform: translateX(-50%); /* Centers the indicator horizontally relative to its position */
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.3); /* Soft shadow for a floating effect */
    transition: left 0.5s ease-out; /* Smooth transition for when its position changes */
}
/* Styling for quiz options on hover */
.quiz-option:hover {
    background-color: rgba(255, 255, 255, 0.3); /* Slightly brighter background on hover */
}
```

```
/* Styling for selected quiz options */
.quiz-option.selected {
    background-color: rgba(255, 255, 255, 0.4); /* More prominent background for selected option */
    border: 2px solid #ffffff; /* White border to clearly indicate selection */
}
```

Appendix B: Screenshots

Folder Substructure and Code Snippets