ASHESI UNIVERSITY

Web Tech Final Project | SatTrack Documentation

Submission Links

GitHub Repository: https://github.com/marzafiee/SatTrack

Live Demo: http://sattrack.42web.io/

Eyra Inez Anne-Marie Agbenu (47152026)

CS341_B - Web Technologies

Kwadwo Osafo-Marfo

December 19, 2025

SatTrack

SatTrack is a lightweight full-stack web application for tracking satellites above your sky, visualizing orbits in 3D, and sharing community observations. It combines accurate orbital data (TLE), server-side pass predictions, and a collaborative observations feed with likes, replies, and nested comments.

Users:

- Space enthusiasts and amateur astronomers interested in satellite observation
- Casual observers curious about satellites passing overhead
- Stargazers who want location-specific predictions
- People wanting to understand what's above them

Problem:

- Difficulty tracking satellites overhead in real time
- Lack of personalized, location-based pass predictions
- Hard to visualize satellite orbits in 3D
- No centralized way to manage a satellite watchlist
- Missing community features to share observations and experiences

Why SatTrack:

- Provides location-specific pass predictions using real TLE data
- Interactive 3D visualization (Cesium) for orbit understanding
- Personal watchlist to track favorite satellites
- Community features for sharing observations and connecting with others
- Accessible interface requiring no specialized tools

Features

- SatTrack offers an interactive 3D globe powered by Cesium, featuring satellite markers that enable users to visualize satellite positions in real-time. Users can maintain a

personal watchlist, adding or removing satellites, and share or import watchlists with others. The application provides pass predictions tailored to your specific location using server-side propagation calculations.

- The platform includes a collaborative observations feed where users can create public posts, like observations, and engage in nested replies and comments.
- The system features CSRF protection and session security to ensure user safety. Server-side geocoding is implemented using Nominatim, allowing users to change their location directly from the user interface. The application features a responsive dashboard design and includes a helpful guide modal to assist new users in getting started quickly.

Core Web Pages

➔ Landing / Home (index.php) - Project landing page with quick actions including project introduction, login and registration options, and demo link.

➔ Dashboard (dashboard.php) - User dashboard featuring the interactive 3D globe, personal watchlist, upcoming satellite passes, and community observations feed.

➔ Login (login.php) / Register (register.php) - User authentication pages for logging in and creating new accounts.

➔ Observations (observations.php) - Community feed where users can post observations, like posts, and reply with nested comments support.

➔ Pass Predictions (passes.php) - Server-side pass predictions personalized to user location and watchlist satellites.

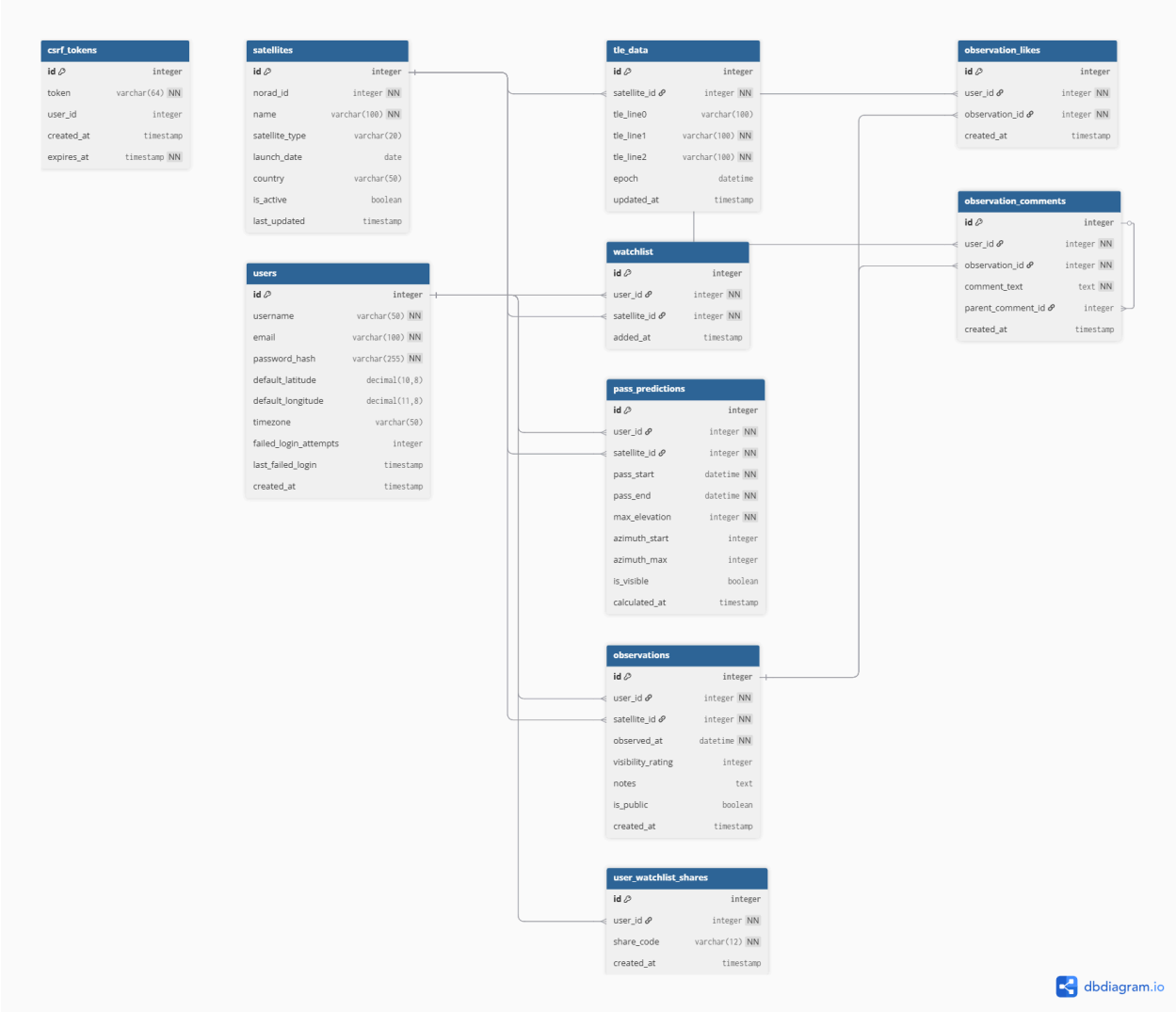
➔ Profile (profile.php) - Account management interface for updating account details, location settings, and watchlist import/export functionality.

➔ Watchlist API (api/add_watchlist.php, api/remove_watchlist.php) - AJAX endpoints that handle adding and removing satellites from a personal watchlist.

➔ About (about.php) / Help (help.php) - Project information pages with guides and UI help documentation.

➔ Developer/Test (test.php) - Miscellaneous developer and testing utilities (not part of the user-facing interface).

Database Design - ERD Diagram



Tech Stack

- The frontend structure and design are built using HTML5, CSS3, and Tailwind CSS via CDN, as well as CesiumJS for 3D visualization and satellite.js for orbital calculations. Client-side interactivity is handled through Vanilla JavaScript without requiring additional frameworks.
- The backend logic and data handling are powered by PHP with MySQLi for database connectivity. MySQL serves as the database management system, with migrations stored in the migrations directory and the main schema defined in database/schema.sql.

- Version control and collaboration are managed through Git and GitHub. Additional libraries include Three.js for enhanced visuals, Nominatim for geocoding services, and N2YO/CelesTrak APIs for seeding satellite data.

System Flow Diagram

```
User Browser (Frontend)
    ↓ HTTP/AJAX Requests
PHP Backend (Application Layer)
    ↓ SQL Queries via PDO
MySQL Database (Data Layer)
    ↓ TLE Data
External APIs (CelesTrak, N2YO)
```

Requirements

- The application requires PHP version 8 or higher, a MySQL database, and a local web server. XAMPP or Apache is recommended for Windows environments.
- Composer is not required for core application code, but may be helpful for additional tooling and development utilities.

Installation Guide

Step 1: Clone the Repository

- Begin by cloning the SatTrack repository from GitHub using the command line.

Step 2: Create and Configure Database

- Create a new database and apply the necessary migrations.
- You can import the schema.sql file located in the database folder.
- Using MySQL CLI or phpMyAdmin, open your database management tool, create a new database (for example, named "sattrack"), and import the .sql file from the /database folder.
- After importing the base schema, run any additional migration SQL files located in the migrations directory if needed.

Step 3: Configure Environment Variables

- Copy the example environment configuration file and set the appropriate values. Create a .env file in the sattrack directory with the following configuration:
    - Database host should be set to localhost.
    - Set your database name (such as "sattrack"). Configure your database user (typically "root" for local development).
    - Set your database password (leave blank if none is set locally).
    - Add your N2YO API key if you plan to use their services. Include your Cesium access token for map visualization.

Step 4: Seed Satellite Data (Critical)

- For development purposes, you can use the seed_satellites.php script located in the database directory to import a sample dataset of satellite information.

Step 5: Run the Project

- Start your local web server, such as XAMPP or WAMP.
- Once the server is running, visit the application in your web browser at http://localhost/sattrack/ or at the path configured by your web server.

Database Migrations and Local Changes

- The main database schema files are located in database/schema.sql.
- The migrations directory contains SQL files for incremental database updates, including scripts to add features like likes and comments, parent comment support, and other enhancements.
- For small changes or testing purposes, you can use phpMyAdmin or the MySQL command line interface to apply schema updates and migrations directly to your local database.

API Endpoints

- The application uses several API endpoints that are called by the front-end JavaScript. These endpoints are located in the api directory and handle various operations.

- The get_tle_data.php endpoint fetches TLE (Two-Line Element) data for satellites in the user's watchlist. The add_watchlist.php and remove_watchlist.php endpoints enable users to add or remove satellites from their personal watchlist.
- The update_location.php endpoint resolves location information using the Nominatim service and saves latitude and longitude coordinates. The add_observation.php endpoint creates new observations and is protected by CSRF (Cross-Site Request Forgery) tokens.
- Comment functionality is handled through multiple endpoints. add_comment.php allows users to add comments or replies to observations and supports the parent_comment_id parameter for nested threading. Meanwhile, get_comments.php retrieves nested comments for a specific observation.
- The toggle_like.php endpoint manages the liking and unliking of observations.
- All write endpoints require user authentication and include CSRF token validation. CSRF tokens are included as hidden input fields in forms. After successful AJAX actions, the server returns a new CSRF token (csrf_new), which the client automatically updates to prevent stale-token errors.

Front-end Behavior and User Experience

- The observations feed supports nested replies with unbounded depth, allowing for complex conversation threads. Updates to comments and likes appear in place without requiring page refreshes.
- The interactive globe displays colored markers for satellites even when TLE data is temporarily unavailable, using placeholder markers to maintain visual continuity. When TLE data is available, the application renders complete orbital paths for enhanced visualization.
- The sidebar help button remains pinned to the bottom of the watchlist panel, regardless of the number of satellites displayed, ensuring users can always access assistance when needed.

Security Notes

- The application implements several security measures to protect user data and prevent common vulnerabilities. Sessions are configured with HttpOnly flags to prevent JavaScript access, SameSite=Strict to prevent cross-site request forgery, and Secure flags when HTTPS is available to ensure encryption in transit.
- CSRF tokens are stored server-side in the csrf_tokens database table. A session-based fallback is implemented if database writes fail, preventing "invalid CSRF token" errors during development. For production deployments, the application should be served over HTTPS with all secure cookie flags properly enabled.

Testing and Troubleshooting

CSRF Token Issues

- If you encounter "invalid csrf token" errors when submitting forms, first ensure that PHP sessions are working correctly by verifying that session cookies are present in your browser. Confirm that the database is reachable and that the csrf_tokens table exists. The system includes a fallback mechanism that stores tokens in the session if database insertion fails.
- After successful AJAX write operations, the server returns a new CSRF token (csrf_new), which the client automatically updates.

Empty Globe Display

- If the 3D globe appears empty without any satellites, confirm that the get_tle_data.php endpoint is returning satellite data for the satellites in your watchlist.
- Note that placeholder markers are displayed even when TLE data is missing, but complete orbital paths require valid TLE data to render properly.