Problem Statement: Interstellar Asteroid Tracker & Risk Analyser

Title:

Cosmic Watch: A Full-Stack Platform for Real-Time Near-Earth Object (NEO) Monitoring

Problem Statement:

In the vastness of space, thousands of Near-Earth Objects (NEOs) pass by our planet daily. While space agencies like NASA track these meticulously, the raw data remains inaccessible or difficult to interpret for the general public, local researchers, and amateur astronomers. There is a lack of localised, user-friendly tools that translate complex trajectory data into understandable risk assessments and visual alerts.

Challenge:

Build a full-stack web platform that fetches live asteroid data from space agency APIs to provide a comprehensive monitoring dashboard. The platform should allow users to track specific objects, understand their potential impact risks, and receive automated alerts based on proximity.

The platform should feature:

- User Authentication & Verification: Secure login for researchers and enthusiasts to save specific "watched" asteroids and set custom alert parameters.
- Real-Time Data Feed: Integration with the NASA NeoWs API to display current asteroids, their velocity, and distance from Earth.
- Risk Analysis Engine: A system that categorises asteroids by "Hazardous" status, diameter, and missed distance to provide a clear risk score.
- Alert & Notification System: Built-in scheduling to notify users of upcoming "close approach" events via the dashboard.
- Containerised Deployment: Deployment ready via Docker, including a docker-compose.yml file to orchestrate the frontend, backend, and database**.**

Bonus Points For:

- 3D Visualisation: An optional interactive 3D view using Three.js or Babylon.js to show the asteroid's orbit relative to Earth.
- Real-time Chat: Capability for users to discuss specific asteroids in a live community thread.

Objective:

Create a robust, scalable, and user-friendly full-stack web application that simplifies complex space data, fostering scientific curiosity and global safety awareness.

General Rules

- All code must be developed during the hackathon timeframe.
- Plagiarism or code copying is strictly prohibited; teams must submit an AI-LOG.md detailing how LLMs were used to assist rather than replace original coding efforts.
- Final submissions should be uploaded to GitHub before the deadline.
- Teams must present a working demo to the judges.
- Postman Collection: A fully documented Postman Collection file included in the repository to test all backend endpoints (Feed, Lookup, and User Profiles).

Development Guidelines

- Tech Stack: Any modern stack (Node.js, Python/FastAPI, React, Next.js, etc.) is allowed.
- Architecture: Backend and frontend must be clearly separated.
- Security: Follow best practices (e.g., hashed passwords, JWT, HTTPS).
- DevOps: Use Git for version control with meaningful commit messages

  .

Evaluation Criteria:

Criteria	Marks	Focus Point
API & Data Architecture	25	Efficiency in handling NASA datasets, backend processing, and RESTful endpoint design.
Full-Stack Implementation	25	Quality of the end-to-end flow from database management to frontend integration.
Docker & Deployment	20	Quality of containerization (Multi-stage builds) and ease of setup via Docker Compose.
Postman API Documentation	10	Completeness of the Postman collection, including environment variables and test cases.
UI/UX Design (Space Theme)	10	Aesthetic appeal, immersive dark-mode visuals, and responsive layout.
3D Graphics (Bonus)	5	Accurate rendering of orbital paths and mathematical scaling in a 3D space.
Real-time Chat(Bonus)	5	Implementation of WebSockets or Socket.io for live community interaction.
Total	**100**	