Tech-documentation

Spacehound AI (vers 1.0)

SpaceHound — это AI-платформа для анализа орбитального трафика: оценивает риск сближений/столкновений спутника, находит “опасные зоны” по высоте и предлагает более безопасные параметры орбиты.

Team members:
Otan Alissultan - 11th grade student of NIS Abay Shymkent
Abzelbek Nurkeldi - 11th grade student of NIS Abay Shymkent
Amanbek Alzhan - 11th grade student of NIS Abay Shymkent
Smail Askhat - 11th grade student of NIS Qaratau Shymkent

Shymkent-2026

Содержание

1.Описание

SpaceHound — AI-платформа для предзапусковой оценки риска орбиты и выбора более безопасных параметров для спутников (особенно CubeSat). Система анализирует высоту и наклонение, выявляет “опасные зоны” перегруженности и предлагает реальные альтернативы орбиты с объяснением причин.

2.Цели и задачи системы

Цель системы SpaceHound — предоставить предзапусковой инструмент поддержки принятия решений для выбора более безопасных параметров орбиты спутника (особенно CubeSat), снижая вероятность опасных сближений и попадания в перегруженные высотные “shells”.

Задачи системы:

1. Принимать параметры миссии (высота, наклонение, дата запуска, длительность) и оценивать риск.
2. Определять перегруженность орбитального региона и “опасные зоны” по высоте (пики плотности).
3. Генерировать реальные альтернативы орбиты (не случайные) и ранжировать их по риску.
4. Выдавать объяснение и текстовую рекомендацию, зависящую от причин риска (case-based).
5. Визуализировать ситуацию в 3D: спутники, треки, фокус, слои и оболочки орбит.

3.Функциональные требования

Основные функции (must-have):

- Анализ риска для введённых параметров миссии.
- Нормализованный risk score (0..1) и категориальный уровень риска (LOW/MEDIUM/HIGH).
- Определение “опасных зон” по высоте: список диапазонов (band), peak altitude, peak density.
- Генерация top-3 альтернатив орбиты с объясняющими notes и отметкой попадания в опасную зону.
- Текстовая рекомендация (не рандом) на основе факторов: congestion, relative velocity, высота, наклонение, risk level.
- 3D-трекер спутников: отображение 200–500 объектов, цвет по орбитальным классам, поиск по NORAD ID, орбитальный трек на прогноз.
- API на GEMINI API key, CORS для интеграции с веб-сайтом.

4. Архитектура системы

SpaceHound реализован по архитектуре “Frontend + API Backend”:

Frontend (Web UI):

Статический сайт (HTML/CSS/JS), разворачивается на GitHub Pages. 3D-рендеринг и пропагация орбит выполняются в браузере (Three.js +

satellite.js).

Backend (FastAPI):

JSON API, реализует риск-оценку, генерацию рекомендаций, вычисление опасных зон.
Подключает ML-модель (joblib/pkl) для прогнозирования риска.

5. Модель данных

Система оперирует следующими сущностями:

MissionInput (ввод миссии):

- `altitude_km` (float) — высота орбиты
- `inclination_deg` (float) — наклонение
- `launch_date` (string, optional) — дата запуска
- `mission_days` (int, optional) — длительность миссии

RiskResult (результат анализа):

- `risk_score` (float, 0..1) — нормализованный риск
- `risk_level` (string) — LOW / MEDIUM / HIGH
- `main_reasons` (list[string]) — ключевые факторы риска

- `recommendation_text` (string) — текстовая рекомендация по ситуации
- `top_options` (list[OrbitOption]) — топ-3 альтернативы

OrbitOption (вариант орбиты):

- `altitude` (float)
- `inclination` (float)
- `risk_score` (float, 0..1)
- `note` (string) — объяснение, почему вариант лучше/хуже
- `danger_zone` (bool) — попадает ли вариант в опасную зону

DangerZone (опасные зоны):

- `band_km` ([min, max]) — диапазон высот
- `peak_altitude_km` (float) — высота пика
- `peak_density` (float, 0..1) — нормализованная плотность

6. Алгоритмы

В системе используются следующие алгоритмические блоки:

1. **Оценка перегруженности (congestion level):**
   На основе распределения объектов/спутников по высоте (или упрощённой модели плотности) строится оценка плотности в окрестности заданной высоты. Используется для определения “опасных” shell’ов.

2. **Оценка относительной скорости (relative velocity, упрощённо):**
   Рассчитывается приближённый показатель “насколько часто” выбранные параметры пересекаются с потоками объектов на близких высотах/наклонениях. (В демо-версии допускается упрощение без ковариаций и CDM.)

3. **ML-прогноз риска:**
   ML-модель получает набор фич (altitude, inclination, congestion, relative velocity) и выдаёт риск. Затем риск приводится к нормализованному виду 0..1 (см. ниже).

4. **Определение опасных зон (density peaks):**
   Строится профиль плотности по высоте: `density(alt_km)`.
   Далее ищутся локальные пики и формируются диапазоны (band) вокруг пиков, которые считаются “опасными зонами”.
   Возвращаются как список `dangerous_zones`.

Генератор рекомендаций (не рандом)

Рекомендации формируются **детерминированно**, на основе причин риска. Система не выдаёт случайные советы, а объясняет конкретно, какие факторы повышают риск и что менять.

Правила (пример логики):

- Если `congestion` высокий → совет: сместить высоту в ближайший “чистый” диапазон (например, ±20–80 км), избегая band опасных зон.

- Если `relative_velocity` высокий → совет: скорректировать наклонение на небольшой диапазон (например, 0.5–3°) для уменьшения пересечений плоскостей (если миссия позволяет).

- Если `risk_level = HIGH` и обнаружена `input_danger_zone` → совет: уйти из диапазона опасной зоны + указать ближайшие безопасные высоты.

- Если `risk_level = MEDIUM` → совет: допустимо, но рекомендуется мониторинг, запас ΔV/манёвров, осторожный выбор даты/времени развёртывания.

- Если `risk_level = LOW` → подтверждение выбора + best practices (пассивное сведение, контроль TLE, условия эксплуатации).

Итог: текст рекомендации зависит от конкретной комбинации факторов и не использует случайные варианты.

7. Frontend (Web UI)

1. Landing / About: ознакомительная страница с идеей проекта, рынком, ценностью и навигацией.

2. Risk Analyzer: форма параметров миссии + вывод результатов:

    - риск (%), уровень риска
    - причины
    - текстовая рекомендация
    - топ-3 альтернативы
    - опасные зоны по высоте
    - мини-график “Risk vs Altitude”

Frontend обращается к backend через Fetch API; адрес backend задаётся переменной (например `localStorage.SPACEHOUND_API`) или конфигом.

8. Тестирование

Цели тестирования:

- корректность расчёта risk_score и его нормализации (0..1)
- стабильность генерации top_options (не дублируются, действительно лучше)
- корректность выявления dangerous_zones и попадания в input_danger_zone
- работоспособность API (status codes, schema)
- корректность фронтенда (все блоки отображаются, запросы уходят)

Типы тестов:

- Unit tests: нормализация риска, детектор пиков, генератор рекомендаций
- API tests: запросы к `/recommend_orbit`, `/danger_zones`, `/report.pdf`
- UI tests (ручные): ввод параметров, отображение, ошибки сети, пустые ответы

9. Безопасность и ограничения

- Никаких API ключей на фронтенде (GitHub Pages виден всем).
- Если используется Gemini/другие LLM — ключ хранится только в backend env.
- Ограничение CORS на проде (разрешить только домен сайта).
- Валидация входных данных (диапазоны высоты/наклонения).
- Логирование без чувствительных данных.

Ограничения текущей демо-версии:

- Риск является оценкой/скорингом и не заменяет официальные CDM/SSA сервисы.
- Упрощённые оценки congestion/relative velocity без ковариаций и точной вероятности столкновения (PoC/Prototype).
- Точность зависит от данных/модели и требует расширения датасета для прод-уровня.

10. Appendix

ссылка на сайт - https://glamzx.github.io/SpaceHound/app.html