Project Requirements Document (PRD): Planet Stories

Team member: Andrew Zhang, Junjie Liu, Michael Wu, Shiyuan Wang, Yixuan Wong

1. Project Vision & Objective

1.1. Project Overview

Planet Stories will be an interactive web portal that ingests, enriches, and displays stories from the **Planet Data API**. We will enhance these stories by applying backend analysis, including geospatial categorization, content-based tagging, and AI-generated captions using Visual Language Models (VLLMs).

1.2. Vision & Mission

The Earth is beautiful, so we want people to see it! Planet.com offers so many stories that capture the change in a particular area. However, these stories are usually provided without much context or description. We aim to make them more accessible to the general audience! This project would allow the general audience to see the change in an area that they are interested in, whether it be their hometown, a place they are moving to, or their dream vacation destination.

1.3. Core Objective

To build and deploy a novel, interactive web portal that ingests stories from the Planet Data API, enriches them with AI-generated context (tags, captions), and provides an advanced, story-based filtering interface. The application's UI/UX will **maintain visual cohesion with Planet's existing brand identity** (e.g., color palette, navigation controls).

2. System Architecture

The application will be built on a modern, decoupled architecture:

- **Frontend:** React, using components from Shadcn/ui for a polished interface. Recharts for data visualization and Zustand for state management.
- **Mapping: Leaflet** for displaying and interacting with geospatial data.

- **Backend:** A Python API built with the FastAPI framework to serve data to the frontend, acting as an **enrichment layer and wrapper for the Planet Data API**.
- **AI/ML:** Server-side calls to pre-trained VLLM APIs (e.g., Gemini, GPT-4o) for image analysis, captioning, and tagging.
- **Database:** PostgreSQL with the PostGIS extension to enable efficient geospatial querying and storage of all story **metadata**. Raw image files will not be stored; the database will hold links to the images.
- **DevOps:** GitHub Actions for Continuous Integration (CI) and to run the scheduled daily data ingestion pipeline.

3. User Stories

ID	Persona	Goal
US-1	Researcher	I want to select two images from different dates and view them side-by-side to directly compare specific changes.
US-2	Resident	I want to select an area I'm interested in on a map and see the available stories of change for that area.
US-3	Geologist	I want to filter and find all stories tagged with "volcano" to survey volcanic activity.
US-4	Scientist	I want to quickly find all images of the Amazon rainforest from October 2024 to quantify deforestation.
US-5	Researcher	I want to view a story's full content, its AI-generated caption, and its map location to evaluate its geographic context.
US-6	Teacher	I want to show my students stories of major environmental events (e.g., wildfires, floods) to help them visually understand their impact.
US-7	Traveler	I want to browse stories by location to see how places I've visited have changed over time.
US-8	Developer	I want to access an API endpoint that returns story metadata to build custom visualizations.

3.2. Detailed Use Case Scenarios

Use Case 1: Viewing Hometown Change (Supports US-2, US-7)

- Actor: A former resident.
- Goal: To see the changes that occurred in the area where they used to live.
- Preconditions: The user can locate their hometown on a map.
- Main Flow:
    1. The user navigates to the Planet Stories web portal.
    2. The user locates their hometown on the map interface, either by panning/zooming or using a search bar (if available).
    3. The user clicks on the map or draws a small box to define their area of interest.
    4. The system filters and displays story pins in that area.
    5. The user clicks a pin for a story that interests them.
    6. The system opens a detail view, showing the story, its images, and AI-generated context.
    7. The user can jump between different timestamps (images) to observe the changes.

Use Case 2: Developer API Access (Supports US-8)

- Actor: Software Developer.
- Goal: Retrieve story data and imagery through an API to build new applications or visualizations.
- Preconditions:
    1. The developer has valid API access credentials (if required).
    2. The Planet Stories backend exposes a documented public API endpoint.
- Main Flow:
    1. (If auth is implemented) The developer obtains an API key from a developer portal.
    2. The developer sends a GET request to /api/stories to retrieve available stories.
    3. The developer appends query parameters (e.g., ?bbox=..., ?tag=volcano, ?start_date=...) to filter the results.
    4. The system returns a JSON response containing story metadata (title, description, date, tags, location, image links).
    5. The system also includes the AI-generated captions in the JSON response.
    6. The developer integrates this data into their third-party visualization (e.g., a custom dashboard, web app, or research tool).

4. Requirements

4.1. Functional Requirements (FR)

- **FR-1 (Data Ingestion):** The system **must** have a scheduled daily pipeline to ingest new stories (titles, images, dates, locations) from the **Planet Data API**.
- **FR-2 (AI Enrichment):** The ingestion pipeline **must** process each new story with a VLLM to generate:
  - a) A concise, descriptive caption (1-2 sentences).
  - b) A list of at least 5 relevant content tags (e.g., "deforestation," "urban growth," "volcano").
- **FR-3 (Storage):** All story metadata, AI-generated content, and location data **must** be stored in the PostgreSQL/PostGIS database.
- **FR-4 (Map UI):** The frontend **must** display all stories as interactive pins on a **Leaflet** map.
- **FR-5 (Card UI):** The frontend **must** also provide a scrollable grid of "story cards," each displaying a title, thumbnail, and tags.
- **FR-6 (Filtering - Text):** Users **must** be able to filter stories by keyword (searching titles and tags). (Supports US-3)
- **FR-7 (Filtering - Date):** Users **must** be able to filter stories by a specified date range. (Supports US-4)
- **FR-8 (Filtering - Geo):** Users **must** be able to filter stories by selecting a point or drawing a bounding box on the map. (Supports US-2)
- **FR-9 (Detail View):** Clicking a story pin or card **must** open a detail view showing all content, images, and AI-generated metadata. (Supports US-5)
- **FR-10 (Compare View):** The detail view **must** include a "Compare" feature that allows users to select any two images for a side-by-side view. (Supports US-1)
- **FR-11 (API - Stretch Goal):** The system **should** provide a public-facing, read-only GET endpoint (/api/stories) that exposes the enriched metadata, separate from the primary application's internal API. (Supports US-8)

4.2. Non-Functional Requirements (NFR)

- **NFR-1 (Performance - Ingestion):** The daily ingestion pipeline **must** complete in under 4 hours.
- **NFR-2 (Performance - API):** All backend filter queries (text, date, geo) **must** return a JSON response in < 1,000ms.
- **NFR-3 (Performance - Load):** The end-to-end page load time for the main stories dashboard **must** be < 3 seconds.
- **NFR-4 (Usability):** The web application **must** be responsive and usable on common

desktop, tablet, and mobile browsers.
- **NFR-5 (Reliability):** The deployed application **must** maintain 99.9% uptime.
- **NFR-6 (Scalability):** The system **must** support 50 concurrent users without significant performance degradation.

5. Project Milestones

Milestones are defined by key deliverables, not just phases.

- **Milestone 1: Core Pipeline & Read-Only UI**
  - *Deliverable:* A deployed web app that proves the core architecture. It successfully ingests (via the Planet API), stores, and displays the 10 most recent stories in a simple list. The ingestion pipeline (manual trigger) is functional.
- **Milestone 2: AI Enrichment & Map Integration (MVP)**
  - *Deliverable:* The ingestion pipeline now successfully enriches stories with AI-generated captions and tags. The frontend displays these stories as pins on a Leaflet map. The core user experience (US-2, US-5) is functional.
- **Milestone 3: Advanced Filtering**
  - *Deliverable:* The filtering sidebar is fully functional. Users can filter stories by text (US-3), date range (US-4), and map selection (US-2). The application is feature-complete for its primary use cases.
- **Milestone 4: Polish & Final Features**
  - *Deliverable:* The side-by-side image comparison view (US-1) is implemented. The UI is fully polished using Shadcn components and aligned with Planet's visual theme. The project is "demo-ready" and deployed to its final URL.