AtmosInsight – Source Code Analysis (Detailed Engineering Version)

Repository prototype: NASA GIBS web viewer with optional podcast-style narration via Google Cloud Text-to-Speech (TTS). This report analyzes each source file, explains core procedures, runtime flows, API design, and provides testing and deployment guidance.

# 1. Architecture Overview

* Frontend (static): index.html, styles.css, app.js (Leaflet-based viewer, layer/date controls, TTS trigger).
* Backend (Node/Express): server.js exposes POST /narrate for Google TTS and serves static assets.
* Package: package.json defines minimal runtime dependencies and start script.
* Docs: README.md with local run instructions and environment setup.

High-level data flow: User → UI selections → build WMTS URL → Leaflet requests GIBS tiles → (optional) POST /narrate → Google TTS → Base64 MP3 → HTMLAudioElement.

# 2. File-by-File Analysis

## 2.1 index.html (structure & dependencies)

Purpose: Declares UI skeleton, loads Leaflet from CDN, binds controls (layer, date, voice), and bootstraps app.js. Contains a side panel for metadata and narration playback.

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />  
<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>  
<select id="layer">  
 <option value="MODIS\_Terra\_CorrectedReflectance\_TrueColor">MODIS Terra – True Color</option>  
 ...  
</select>  
<input type="date" id="date"/>  
<button id="gen">Generate Narration</button>  
<select id="voice">en-US / es-ES / es-CO ...</select>  
<div id="map"></div>  
<aside id="panel">Layer Info • Tiles URL • Narration • <audio/></aside>  
<script src="./app.js"></script>

Notes: The layer list uses NASA/partner identifiers compatible with GIBS WMTS. The page references styles.css and app.js residing next to index.html.

## 2.2 styles.css (layout & responsiveness)

Purpose: Responsive two-column layout (map + info panel), sticky header with control group, subtle system UI palette, and mobile fallback to single-column.

header{display:flex;gap:16px;align-items:center}  
main{display:grid;grid-template-columns:1fr 380px;gap:8px;height:calc(100vh - 58px)}  
#map{height:100%;width:100%}  
#panel{overflow:auto;background:#fff;border-left:1px solid #e5e7eb;padding:12px}  
@media (max-width:980px){ main{grid-template-columns:1fr} }

Notes: Uses plain CSS without Tailwind; ensures the Leaflet canvas occupies the full viewport height.

## 2.3 app.js (UI logic, GIBS integration, narration trigger)

Purpose: Initializes Leaflet map, composes WMTS URLs for NASA GIBS in EPSG:3857, updates tiles based on user selections, builds short narration text, and requests audio from backend.

function wmtsUrl(layer, date){  
 return `https://gibs.earthdata.nasa.gov/wmts/epsg3857/best/${layer}/default/${date}/GoogleMapsCompatible\_Level9/{z}/{y}/{x}.png`;  
}  
function defaultUTCDate(){  
 const d=new Date(); d.setUTCDate(d.getUTCDate()-1);  
 return `${d.getUTCFullYear()}-${String(d.getUTCMonth()+1).padStart(2,"0")}-${String(d.getUTCDate()).padStart(2,"0")}`;  
}  
const map = L.map("map",{center:[4.711,-74.072], zoom:4, minZoom:1, maxZoom:9});  
let tileLayer = L.tileLayer(wmtsUrl(layer,date),{ attribution:"© NASA EOSDIS GIBS" }).addTo(map);

Key procedures:

* refreshLayer(): recomputes WMTS URL and updates the active TileLayer via setUrl().
* buildNarrative(): composes a short explainer string from layer label + date (input to TTS).
* Event handlers: changes in <select id='layer'> and <input id='date'> call refreshLayer().
* Narration button: POST /narrate { text, voice } → expects JSON { audioContent: base64 } → sets <audio src='data:audio/mp3;base64,...'> and plays.

## 2.4 server.js (Express + Google Cloud TTS)

Purpose: Serves static frontend and exposes POST /narrate that synthesizes MP3 audio using Google Cloud Text-to-Speech. Also maps voice code to language and returns Base64 payload to the client.

const client = new tts.TextToSpeechClient();  
app.post("/narrate", async (req,res)=>{  
 const { text, voice="en-US-Neural2-C" } = req.body || {};  
 const [response] = await client.synthesizeSpeech({  
 input:{ text },  
 voice:{ languageCode: voice.startsWith("es-")?"es-ES":"en-US", name: voice },  
 audioConfig:{ audioEncoding:"MP3", speakingRate:1.02 },  
 });  
 res.json({ audioContent: response.audioContent.toString("base64") });  
});

Notes: Requires GOOGLE\_APPLICATION\_CREDENTIALS to point to a valid service account JSON with the TTS API enabled.

## 2.5 package.json (runtime & deps)

{  
 "name": "atmosinsight-mvp",  
 "scripts": { "start": "node server.js" },  
 "dependencies": {  
 "express": "^4.19.2", "cors": "^2.8.5", "@google-cloud/text-to-speech": "^5.3.0"  
 }  
}

## 2.6 README.md (usage)

Describes local execution steps, credential export, and the WMTS template used by the viewer.

# 3. Runtime Flow & Sequence

User  
 │ selects layer/date/voice  
 ▼  
Frontend (app.js)  
 ├─ build WMTS URL → Leaflet TileLayer.setUrl(...)  
 └─ POST /narrate {text, voice}  
 ▼  
Express server (server.js)  
 └─ Google TTS → Base64 MP3 → JSON {audioContent}  
 ▼  
Browser  
 └─ <audio src="data:audio/mp3;base64,...">.play()

# 4. API Contract – POST /narrate

Request (JSON): { text: string, voice?: string }

Response (200): { audioContent: string(base64 MP3) }

Errors: 400 (missing text), 500 (synthesis failure or credential issues).

Security: Never commit service-account.json; use environment variable and secrets manager.

# 5. Testing Plan

Manual checks:

* Load tiles for recent date (yesterday UTC).
* Switch between MODIS/VIIRS/SMAP/TROPOMI/OPERA layers and verify URL shape and attribution.
* Narration success path (valid credentials) and failure handling (alert + console error).
* Suggested automated tests (headless browser + mock):
* wmtsUrl() builds expected URL for known inputs.
* defaultUTCDate() returns YYYY-MM-DD and ≤ today-1.
* POST /narrate with mock Google client returns base64 payload.

# 6. Performance & Reliability

* Use GIBS caching headers; avoid rapid date toggles (debounce) to reduce tile churn.
* Optionally prefetch adjacent dates for short time-lapse UX.
* Add basic logging and rate limiting on /narrate for robustness.

# 7. Deployment & Configuration

* Single-node: serve static files and /narrate from the same Express instance (port 8787 by default).
* Split deployment: host index.html+app.js+styles.css on static hosting; run server.js separately and point app.js endpoint to that origin (CORS enabled).
* Env: GOOGLE\_APPLICATION\_CREDENTIALS to service account with TTS enabled; outbound internet required for GIBS tiles and Google API.

# 8. Roadmap (Next)

* Internationalization (EN/ES UI strings).
* Time-lapse animation with preloaded frames and export to GIF/MP4.
* Bookmarkable permalinks (layer/date/viewport in URL).
* Analytics events for layer popularity/latency.