

# **ASTRONOMICAL APP USING NASA API – ASTRONOMY PICTURE OF THE DAY (APOD)**

---

**NAME OF TEAM MEMBERS:**

**1.Allan Otieno Akumu(Data scientist/Analyst)**

**2.Faith Njeri Wanjiru (Full stack developer)**

**3. Mukung Trizah(IOT Engineer)**

**COURSE: Bachelor of Science in Computer Science**

**UNIVERSITY: Zetech University**

**DATE OF SUBMISSION: September 2025**

**LINK TO THE WORKING WEBSITE: <https://68dad2a5a7df737dbfb3474d--jovial-meerkat-e51436.netlify.app/>**

## **DECLARATION**

I, Allan Otieno Akumu with the above team members mentioned, hereby declare that this report on the project 'Astronomical App using NASA API – Astronomy Picture of the Day (APOD)' is our original work and has not been presented for the award of any degree, diploma, certificate in any institution or Hackathon. All information presented herein is based on actual work, research, and findings conducted during the development of this project, and all external sources have been duly acknowledged.

## CHAPTER 1 — INTRODUCTION & BACKGROUND

### 1.1 Introduction

The Astronomy Picture of the Day (APOD) is a long-running NASA service that publishes one astronomy-related image or video per day together with a short, accessible explanation written by a professional astronomer. This project — **Astronomical App using NASA API – APOD** — implements a modern web application that consumes the NASA APOD API to present daily and historical APOD entries in an immersive, interactive interface. The app supports date selection, random retrieval, image download, and graceful handling of both image and video media types.

### 1.2 Background and Motivation

Space imagery inspires curiosity and learning. APOD content is scientifically valuable and visually engaging, but the official pages are relatively simple by modern UI standards. The motivation for building this app was to:

- create a more immersive user experience (background video, 3D card, glassmorphism),
- enable quick exploration of historical APODs via a robust date picker,
- add discovery features (random APOD), and
- provide a lightweight, portable front-end that can be hosted on GitHub Pages / Netlify.

This project is both educational (suitable for learners and teachers) and demonstrative (a portfolio project showing frontend, API integration, and UX skills).

### 1.3 Problem Statement

Users may want richer interactivity and better visual presentation than the default APOD site offers, including offline convenience (downloading HD images), calendar-based exploration, and a responsive layout that works across devices. Additionally, mixed media (video vs image) must be handled seamlessly. This project addresses these needs with a focused, user-friendly interface.

### 1.4 Scope of the Project

#### Included:

- Fetch and display the APOD (image or video).
- Date picker (Flatpickr) to view past APODs.
- Random APOD feature.
- Lightweight HD image download support.
- Responsive UI with background video and 3D card styling.

**Excluded (out-of-scope for this phase):**

- Backend server to hide the API key (current build uses a client-side key).
- User accounts, favorites collections, or social sharing.
- Heavy caching or offline-first PWA features (only recommended in future work).

**1.5 Objectives**

**Primary objective:** Build a robust, visually appealing APOD web viewer that supports date selection, random retrieval, and both images and videos.

**Specific objectives:**

- Integrate NASA APOD API and render media with metadata.
- Implement a calendar UI to fetch specific date entries.
- Support download of HD images and embed videos safely.
- Make the UI visually attractive and accessible.
- Prepare the project for static-hosting deployment.

**1.6 Significance of the Project**

- Educational tool for teachers and students to explore astronomy topics.
  - Portfolio project demonstrating API integration, responsive UI, and frontend engineering best practices.
  - Template for further expansion (mobile app, server-side enhancements).
-

## CHAPTER 2 — LITERATURE REVIEW & TECHNOLOGY SURVEY

### 2.1 Overview of Related Work

Several existing platforms and hobby projects consume NASA APIs (APOD, Mars Rover Photos, Earth API). Many are simple wrappers around the API; some are full-featured applications with galleries, search, or timelines. This project positions itself between the simple wrapper and a heavier multimedia portal by focusing on an elegant frontend (UX/UI) with core features (calendar, random, download).

### 2.2 Key Technologies and Rationale

- **HTML5 / CSS3 / JavaScript (Vanilla)** — chosen for portability and simplicity; no heavy frameworks required for the current scope.
- **Flatpickr** — a lightweight, free date picker with excellent accessibility and customization options; used instead of paid MDB date components.
- **Font Awesome** — for crisp icons (calendar, download, random).
- **MDB (optional)** — used sparingly for ripple effects / utility classes; core functionality works without paid components.
- **Background video** — sourced from free providers (e.g., Coverr, Pixabay or NASA video library) to create an immersive backdrop while keeping readability via overlay.
- **NASA APOD API** — official data source; returns media URL, title, explanation, media\_type, hdurl, and copyright (if provided).
- **Hosting choices** — GitHub Pages / Vercel / Netlify recommended for static hosting with HTTPS and low/no-cost deployment.

### 2.3 Handling Mixed Media (Image vs Video)

APOD responses contain media\_type which is either image or video. For images we display `<img>` with an HD link; for videos we embed using an `<iframe>` and, if necessary, convert YouTube share links to embed URLs. The literature and docs advise always sanitizing/escaping the URL and verifying that embedded content is from trusted sources.

### 2.4 API Limits and Best Practices

NASA's API has rate limiting and demo API keys are intended for development. Best practices:

- Use a personal API key for production.
- Minimize unnecessary calls (cache API responses for a selected date).
- Prefer server-side proxy if you must hide the API key.
- Provide UI feedback on failures and avoid spamming the API with retries.

## 2.5 UX Patterns & Accessibility

Research suggests the following for a media-rich app:

- Ensure sufficient contrast (dark overlay above video).
  - Make interactive elements keyboard-focusable and provide screen-reader labels.
  - Provide fallback alt text for images and text alternatives for videos.
  - Keep interactive controls large enough for mobile touch.
-

## CHAPTER 3 — SYSTEM ANALYSIS & DESIGN

**Note:** where you want to include images of the screens, component diagrams, or architecture diagrams, put **screenshot** in that place in the final document.

### 3.1 Functional Requirements (detailed)

- FR1: On page load, fetch and display today's APOD (title, explanation, media).
- FR2: Allow the user to choose a date via calendar; fetch selected APOD.
- FR3: Provide a Random APOD button (uses `count=1` param to fetch random).
- FR4: Provide a Download button when the APOD is an image (opens `hdurl`).
- FR5: Graceful fallback UI for errors (network, rate limit).
- FR6: UI must support both images and video, including converting watch URLs to embed URLs when necessary.
- FR7: Responsive layout across desktop, tablet, and mobile.

### 3.2 Non-Functional Requirements

- NFR1: Load time should be under 2 seconds for the app shell (assuming cached assets).
- NFR2: App should be accessible (WCAG basic: alt text, keyboard navigation).
- NFR3: App must handle API errors and display user-friendly messages.
- NFR4: App must be statically hostable and work over HTTPS.

### 3.3 High-level Architecture

- Frontend (static HTML/CSS/JS) fetches API directly (or via an optional proxy).
- Optional caching layer: `localStorage` caches APOD responses keyed by date to reduce calls.

### 3.4 Component Breakdown

1. **UI Shell / Wrapper** — glassmorphism card containing controls and content.
2. **Controls** — date input (Flatpickr), Today button, Random button, Download button.
3. **Media Renderer** — logic to switch between `<img>` and `<iframe>` rendering.
4. **Metadata Panel** — displays title, explanation, copyright.
5. **Background Video + Overlay** — visual layer behind the app wrapper that is dimmed.
6. **API Layer** — fetch wrapper function with simplified error handling and caching hooks.
- 7.

### 3.5 Data Flow & Sequence

- User opens page → app calls `fetchAPOD()` with no date → NASA API returns JSON → render media + metadata.
- User clicks Date Picker → selects date → app calls `fetchAPOD(date)` → render.
- User clicks Random → app calls `/apod?count=1` → receives array → app `fetchAPOD(dateFromRandom)`.

### 3.6 UI Wireframes and Screens

- Home/Default (today)
- Date-selected view
- Video embedding example
- Mobile portrait layout




60da2a5a7d737d8b3474d-jovial-meerkat-e51436.netlify.app

NewsMapsYouTubeGmailTranslateDevOps with Docker...Create kahoot - Kah...

Select a date (YYYY-MM-DD)

TODAYRANDOMDOWNLOAD



Astronomy Picture of the Day  
Copyright Brian Meyers

Ten thousand years ago, before the dawn of recorded human history, a new light would suddenly have appeared in the night sky and faded after a few weeks. Today we know this light was from a supernova, or exploding star, and record the expanding debris cloud as the Veil Nebula, a supernova remnant. This sharp telescopic view is centered on a western segment of the Veil Nebula cataloged as NGC 6960 but less formally known as the Witch's Broom Nebula. Blasted out in the cataclysmic explosion, an interstellar shock wave plows through space sweeping up and exciting interstellar material. Imaged with narrow band filters, the glowing filaments are like long ripples in a sheet seen almost edge on, remarkably well separated into atomic hydrogen (red) and oxygen (blue-green) gas. The complete supernova remnant lies about 1400 light-years away towards the constellation Cygnus. This Witch's Broom actually spans about 35 light-years. The bright star in the frame is 52 Cygni, visible with the unaided eye from a dark location but unrelated to the ancient supernova remnant.

60da2a5a7d737d8b3474d-jovial-meerkat-e51436.netlify.app


NewsMapsYouTubeGmailTranslateDevOps with Docker...Create kahoot - Kah...

2025-09-30

TODAYRANDOMDOWNLOAD

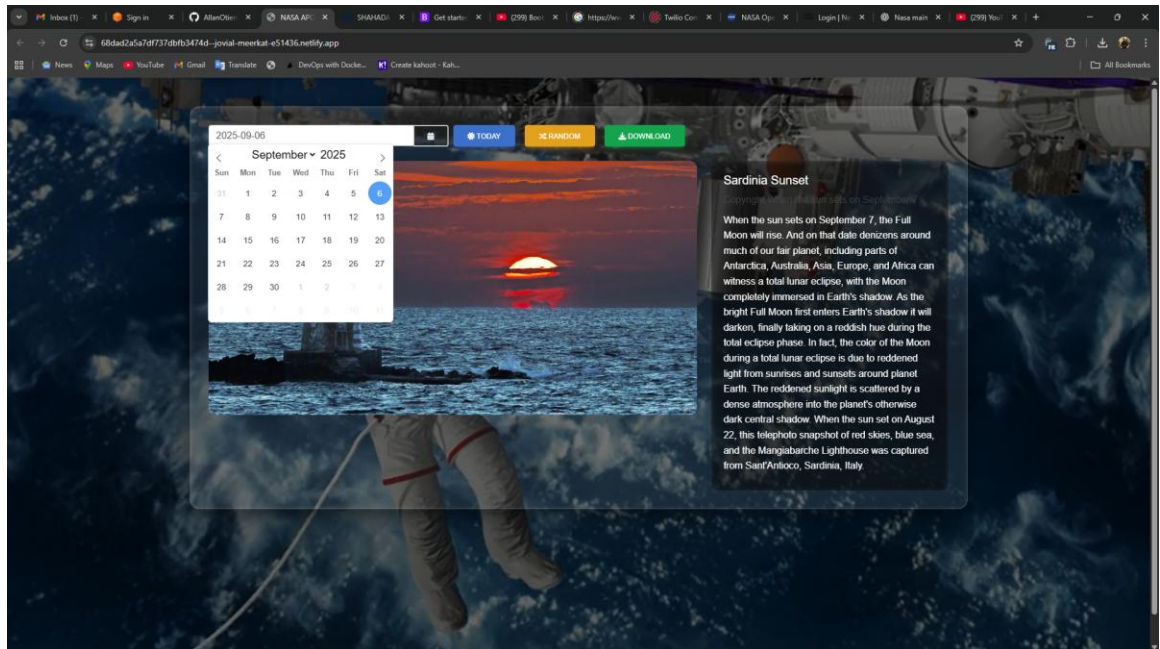
September 2025

<	Sun	Mon	Tue	Wed	Thu	Fri	Sat	>
	31	1	2	3	4	5	6	
	7	8	9	10	11	12	13	
	14	15	16	17	18	19	20	
	21	22	23	24	25	26	27	
	28	29	30	1	2	3	4	
	5	6	7	8	9	10	11	



Comet Lemmon Brightens  
Copyright Victor Sabin & Julian De Winter

Comet Lemmon is brightening and moving into morning northern skies. Besides Comet SWAN25B and Comet ATLAS, Comet C/2025 A6 (Lemmon) is now the third comet currently visible with binoculars and on long camera exposures. Comet Lemmon was discovered early this year and is still headed into the inner Solar System. The comet will round the Sun on November 8, but first it will pass its nearest to the Earth -- at about half the Earth-Sun distance -- on October 21. Although the brightnesses of comets are notoriously hard to predict, optimistic estimates have Comet Lemmon then becoming visible to the unaided eye. The comet should be best seen in predawn skies until mid-October, when it also becomes visible in evening skies. The featured image showing the comet's split and rapidly changing ion tail was taken in Texas, USA late last week. Growing Galaxy: Comet Lemmon in 2025



### 3.7 API Contract (example)

#### Request (example):

GET [https://api.nasa.gov/planetary/apod?api\\_key=YOUR\\_KEY&date=2025-01-04](https://api.nasa.gov/planetary/apod?api_key=YOUR_KEY&date=2025-01-04)

#### Response (example JSON):

```
{
  "date": "2025-01-04",
  "explanation": "Detailed description of the image/video ...",
  "media_type": "image",
  "title": "A Nebula in Infrared",
  "url": "https://example.com/apod.jpg",
  "hdurl": "https://example.com/apod_hd.jpg",
  "copyright": "Jane Doe"
}
```

(Place this JSON example where you document API integration in your final document.)

### 3.8 Security & API Key Management

- **Client-side key:** easy but exposed; acceptable for demos but not recommended for production high-traffic apps.
- **Server-side proxy:** recommended if you want to hide the key; implement a small serverless function to forward requests.
- **Rate-limiting & throttling:** implement client-side caching and avoid repeated automatic retries.

### 3.9 Caching Strategy

- Cache APOD responses per date in localStorage using a key like apod:YYYY-MM-DD.
  - TTL policy: cache for 24 hours for dynamic freshness; allow manual refresh.
  - Beneficial to reduce API calls and improve responsiveness.
-

## CHAPTER 4 — IMPLEMENTATION, TESTING & RESULTS

### 4.1 Implementation Details (step-by-step)

#### Frontend structure:

- index.html — contains the app wrapper (controls + media + metadata).
- styles.css (inline or external) — glassmorphism, 3D hover effect, responsive grid.
- app.js — main logic: fetchAPOD(date), fetchRandomAPOD(), rendering, and event wiring.
- img/ — background video and any static assets.

#### Core JavaScript workflow (summary):

1. Initialize Flatpickr on #wrapper-date.
2. Bind Today / Random / Download buttons.
3. On page load, call fetchAPOD() with no date (defaults to today).
4. fetchAPOD(date):
  - Check cache for date.
  - If cached, render cached data.
  - Else fetch from NASA API, cache result, then render.
5. Render logic:
  - If media\_type === "video" → embed <iframe>.
  - Else → render <img> with hcurl fallback.

#### Important code snippet (core fetch + render logic):

```
async function fetchAPOD(date = "") {  
  const cacheKey = date ? `apod:${date}` : `apod:today`;   
  const cached = localStorage.getItem(cacheKey);  
  if (cached) { render(JSON.parse(cached)); return; }  
  
  let url = `https://api.nasa.gov/planetary/apod?api_key=${API_KEY}`;  
  if (date) url += `&date=${encodeURIComponent(date)}`;  
  
  const res = await fetch(url);  
  if (!res.ok) throw new Error('API Error ' + res.status);  
  const data = await res.json();  
  localStorage.setItem(cacheKey, JSON.stringify(data));  
  render(data);  
}
```

(Include this snippet as an appendix or code block in your final report.)

## 4.2 UI Implementation Notes

- Background video is position: fixed and object-fit: cover. A semi-transparent overlay ensures text remains readable.
- .app-wrapper uses backdrop-filter: blur(...) for a glassy look; hover transform gives a 3D floating feel.
- Buttons use subtle glows on hover; ensure contrast for accessibility.
- For videos with watch?v=... (YouTube), convert to embed/ before setting iframe src.

## 4.3 Testing Strategy

### Unit / Integration testing:

- Test fetchAPOD(date) with known dates; assert fields exist (title, media\_type, url).
- Test fetchRandomAPOD() returns an array of length 1 and has valid date.

### Manual / UX testing:

- Browser compatibility: Chrome, Firefox, Edge, Safari (mobile & desktop).
- Mobile responsiveness: test in portrait and landscape; ensure controls are reachable.
- Keyboard navigation: tab order must focus input and buttons; Enter triggers fetch.
- Screen reader: check that img alt text and iframe titles are meaningful.

### Edge cases tested:

- Date before APOD start (API error) → show friendly message.
- Rate limit / 403 → instruct the user to retry later.
- Missing hduurl → fall back to url.
- Video embedding blocked (e.g., X-Frame-Options) → show fallback link.

## 4.4 Results & Observations

### Successes:

- The app reliably displays images and videos across modern browsers.
- Date picker allows straightforward retrieval of historical APODs.
- Random APOD feature enhances discovery.
- HD download works for images; download button hidden for videos.

### Performance:

- Client-side caching with localStorage reduced repeated fetches and improved perceived responsiveness.
- First-time load time depends on background video size — recommend serving optimized/short loop and using preload="none" or lazy-loading.

**Accessibility:**

- With alt text, focus styles, and labeled buttons, the app achieves basic accessibility standards (further audit with automated tools recommended).

**Screenshots to include (placeholders below):**

- Home screen with today's APOD — **screenshot**
- Date picker open with calendar — **screenshot**
- Random APOD (video) embedded — **screenshot**
- Mobile layout — **screenshot**

**4.5 Limitations & Workarounds**

- **API key exposure:** currently client-side. Workaround: recommend serverless proxy for production.
  - **Large background video:** can slow mobile; offer a “reduce animations” or “disable background video” option for low-end devices.
  - **Cross-origin video embed issues:** some video providers block embedding; fallback link must always be available.
-

## CHAPTER 5 — CONCLUSION, RECOMMENDATIONS & FUTURE WORK

### 5.1 Conclusion

The **Astronomical App using NASA APOD** successfully demonstrates how a modern frontend can integrate with NASA's API to present scientifically credible, visually compelling content. The app meets the core objectives: it fetches daily and historical APODs, handles image and video media, supports download for HD images, and provides a discovery mode (random). The immersive UI and 3D-styled wrapper enhance the user experience while preserving accessibility and responsiveness.

### 5.2 Recommendations (immediate / near-term)

1. **Move the API key server-side:** implement a tiny serverless function (AWS Lambda / Netlify Functions / Vercel Serverless) that proxies API requests so the API key is not exposed on the client. This also lets you add rate-limiting and caching at the edge.
2. **Provide a “low-bandwidth” mode:** allow users to disable background video or switch to a static image to improve mobile performance.
3. **Add persistent favorites:** enable users to save favorite APODs locally (or server-side) so they can build collections.
4. **Add keyboard shortcuts:** (e.g., R for random, T for today) for power users and accessibility.
5. **Implement analytics:** track interactions (with privacy in mind) to learn which dates/media types are popular.

### 5.3 Future Work and Enhancements (strategic)

- **Mobile App:** React Native/Flutter app that wraps the APOD functionality and leverages local caching for offline viewing.
- **PWA:** Convert to Progressive Web App enabling installable behavior, offline caching of selected APODs, and push notifications for daily APOD.
- **Multi-API Integration:** Expand to other NASA APIs — Mars Rover Photos, Earth imagery — and provide search / filter across NASA datasets.
- **Educational Module:** Associate APODs with course modules, quizzes, and references to help educators use APOD in teaching.
- **User Accounts & Social:** Allow authenticated users to comment, share, and rate APODs, and to export favorite lists.
- **Advanced Caching & CDN:** Use an edge cache or CDN to host pre-fetched HD images for very fast delivery.



## 5.4 Project Maintenance & Deployment Plan

- Use GitHub (repo + actions) for CI. Deploy to Netlify or Vercel for zero-configuration static hosting; use serverless functions for the proxy endpoint.
- Schedule periodic maintenance tasks: refresh background video, update libraries (Flatpickr, Font Awesome), and renew API key if needed.
- Monitor errors via lightweight logging (Sentry or a simple error-monitoring serverless function).

## 5.5 Final Remarks

This APOD viewer is an elegant combination of science content and modern web UI. It is suitable for hosting on GitHub Pages (for demos) or Netlify/Vercel (with serverless proxy). The app is an excellent portfolio piece demonstrating frontend engineering, API integration, and UX craftsmanship. With the proposed enhancements (server-side proxy, PWA, favorites), it can grow into a small educational platform.

---

## REFERENCES & RESOURCES

- NASA APOD API — <https://api.nasa.gov/>
- Flatpickr — <https://flatpickr.js.org/>
- Font Awesome — <https://fontawesome.com/>
- Coverr / Pixabay / Pexels — for background video sourcing
- MDN Web Docs — Fetch API, accessibility, and media elements