

AstroExpert: A One-Stop Platform for AI-Driven Astronomical Observation and Smart Commerce

Jayesh Sonawane¹, Vedvrat Bedekar², Yash Gangurde³, Prof. S. L. Dawkhari⁴

¹Department of Information Technology, Sinhgad College of Engineering, Pune-41

²Department of Information Technology, Sinhgad College of Engineering, Pune-41

³Department of Information Technology, Sinhgad College of Engineering, Pune-41

⁴Department of Information Technology, Sinhgad College of Engineering, Pune-41

ABSTRACT - In the modern landscape of amateur astronomy, enthusiasts frequently struggle with fragmented digital resources, unpredictable environmental conditions, and disconnected equipment marketplaces. Planning a successful observation session currently requires navigating multiple independent platforms for weather forecasts, star maps, and gear procurement. This paper presents the development of "Astro Expert," a comprehensive, AI-enhanced web application built to centralize and automate these planning phases. To deliver highly accurate viewing forecasts, the system integrates the OpenMeteo API and local celestial libraries alongside the NASA APOD API for daily educational content. The core innovation of the platform is a custom predictive engine powered by a TensorFlow deep learning model. Utilizing a three-layer dense neural network trained over 50 epochs on 10,000 historical meteorological records, the model analyzes humidity, temperature, and dew points to forecast optimal stargazing timeslots. Furthermore, the platform features a domain-restricted conversational interface powered by the Gemini API, which translates raw predictive data into easily understandable observation alerts. Finally, an integrated e-commerce module ("Smart Astro-Cart") securely processes transactions and recommends specialized equipment tailored to the user's predicted local weather. This paper outlines the system's architecture, machine learning methodology, performance evaluations, and its effectiveness in translating complex atmospheric data into actionable insights for the enthusiast community.

Keywords: Artificial Intelligence, Amateur Astronomy, Deep Learning, Predictive Modeling, Natural Language Processing, Web Application, E-commerce.

1. INTRODUCTION

The practice of amateur astronomy is heavily dependent on unpredictable environmental variables. Currently, enthusiasts lack a centralized ecosystem to manage their hobby. They are forced to cross-reference multiple, disconnected applications to track weather patterns, identify celestial positions, and purchase appropriate viewing equipment. Because traditional weather applications lack domain-specific predictive intelligence for stargazing, users often struggle to make quick, informed decisions regarding when and where to observe.

The primary objective of this project is to eliminate this friction by developing a unified web platform that translates complex meteorological data into practical, actionable insights. By leveraging advanced deep learning (DL) and Natural Language

Processing (NLP), the "Astro Expert" platform simplifies the observation planning process.

Our approach delivers a fully functional web application equipped with a robust TensorFlow-based predictive engine. This engine evaluates real-time and historical weather data to forecast precise stargazing time windows. To bridge the gap between raw data and user experience, a domain-restricted conversational AI answers astronomy-related queries and provides smart alerts. Coupled with an integrated, secure e-commerce storefront that recommends gear based on local weather predictions, this project aims to make astronomy significantly more accessible and enjoyable for both beginners and seasoned enthusiasts.

2. LITERATURE SURVEY

The foundation of this project is built upon recent academic explorations into automated web development, AI integration in education, and system quality metrics.

Research by Jovanić (2024) and Bekmanova (2024) highlights the shift toward Server-Side Rendering (SSR) frameworks to generate responsive and SEO-optimized web content automatically. While Jovanić focused on AI-driven content generation for web builders, and Bekmanova emphasized Nuxt.js implementations, both studies primarily resulted in prototype systems rather than fully commercialized, full-stack applications capable of handling complex AI and database integrations. Similarly, Nannat (2024) explored website creation using closed ecosystems like the Odoo builder. While effective for basic e-commerce, these closed systems lack the flexibility required to integrate custom deep learning models or external real-time APIs (like OpenMeteo or NASA APOD).

In the context of educational technology, Aydınli (2025) strongly advocates for the integration of modern technologies—specifically Artificial Intelligence—into space and astronomy education to overcome traditional teaching barriers. However, this research focuses largely on theoretical pedagogical frameworks rather than the practical engineering required to process live atmospheric data for real-world stargazing. Finally, Ag (2025) conducted a comprehensive bibliometric analysis of website quality over the last two decades. While this provided a strong baseline for the Non-Functional Requirements (NFRs) targeted in our project (such as response time and uptime), it emphasized quantitative metadata over the technical implementation of secure, AI-driven architectures.

The existing literature confirms the necessity for intelligent, adaptive platforms. However, there is a clear gap regarding an open-source, fully integrated ecosystem that combines deep learning weather prediction, conversational AI, and personalized e-commerce into a single astronomy platform. Astro Expert was developed to specifically bridge this gap.

3.SYSTEM REQUIREMENTS AND METHODOLOGY

The development lifecycle followed an agile, modular approach utilizing a modern technology stack. The frontend was constructed using React, JavaScript, and HTML5/CSS3 to ensure 100% mobile responsiveness. The backend operations and machine learning models were powered by Python, utilizing TensorFlow for the predictive engine and the Gemini API for natural language processing.

3.1 Functional and Non-Functional Requirements

To ensure a robust user experience, the system was designed around strict requirements. Key Functional Requirements (FRs) included:

1. Fetching real-time weather via the OpenMeteo API.
2. Calculating environmental quality scores using the TensorFlow model to classify future timeslots.
3. Restricting the Gemini chatbot to strictly astronomy-related topics.
4. Translating raw machine learning outputs into readable user alerts.
5. Processing secure, tokenized e-commerce payments without storing raw card data.

Crucial Non-Functional Requirements (NFRs) dictated that the platform must maintain a response time of less than 2.0 seconds, achieve 99.9% uptime, utilize high-level encryption for payments, and ensure the TensorFlow predictions maintain an accuracy threshold above 85%.

3.2 Predictive Model Development

The core predictive engine was trained on a meticulously calibrated dataset comprising 10,000 historical meteorological records. To prepare this data for deep learning, a rigorous pre-processing methodology was applied. Continuous variables such as humidity, wind velocity, and temperature were normalized using min-max scaling to prevent large-magnitude features from dominating the learning weights. The dataset was subsequently partitioned into an 80/20 train-test split to strictly evaluate the model's performance on unseen atmospheric anomalies.

The architecture consists of a three-layer dense neural network built with TensorFlow. To prevent data interference and ensure precise mathematical boundary classification (e.g., specific dew point and humidity thresholds), the predictive logic was structured utilizing 30 isolated models. These models were trained over 50 epochs using the Adam optimizer and categorical cross-entropy loss functions to classify environmental quality scores and evaluate visibility continuity for extended viewing sessions.

3.3 System Integration Strategy

To manage the complexity of connecting multiple external APIs with internal machine learning engines, the platform utilizes a unidirectional, edge-computing data flow. Real-time telemetry is ingested via the OpenMeteo API and buffered locally. Instead of relying solely on cloud processing, the platform utilizes TensorFlow.js to apply static, pre-trained weight files to the telemetry directly at the edge layer. The resulting numerical "Seeing Condition Score" is then passed sequentially to the Gemini API, which translates the hard metrics into a natural language summary. This sequential integration ensures that the UI dashboard only renders once all predictive and conversational data is fully processed and synchronized.

4.SYSTEM ARCHITECTURE AND FUNCTIONALITIES

4.1 System Architecture

The Astro Expert platform is engineered upon a robust, multi-tiered edge-computing architecture. This modular design is strategically chosen to ensure high responsiveness, secure data handling, and seamless integration of complex machine learning models directly within the application environment. By decoupling the user interface from state management, artificial intelligence processing, and cloud infrastructure, the system maintains strict separation of concerns, which is critical for long-term scalability and maintainability. As illustrated in Figure 1, the architecture is categorized into five distinct functional layers.

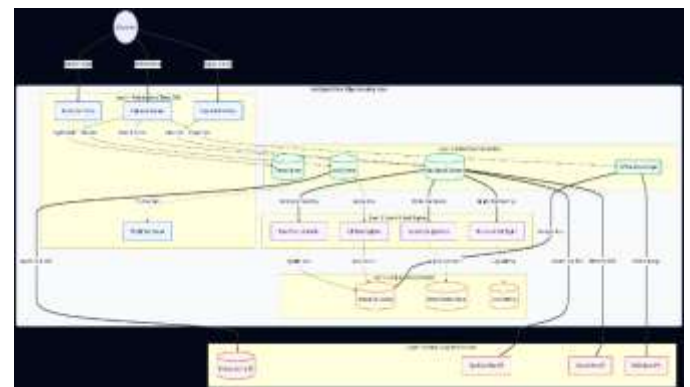


Fig -1: High-level system architecture of the Astro Expert platform.

Layer 1: Presentation & Virtual DOM: Built using modern frontend frameworks (React), this layer manages the dynamic user interface. It renders key visual components like the AI Assistant Menu, Dashboard Interface, and Gear Panel Overlay, capturing user inputs to pass down to state controllers.

Layer 2: Global State & UI Controllers: Acting as the application's central hub, this layer manages global context. It utilizes specific providers (Theme, Auth, and Data Spatial Contexts) to synchronize data across the UI and includes an Offline Sync Manager to coordinate data flow during network drops.

Layer 3: Core AI & Math Engines: The computational heart of the platform. It executes primary analytical tasks using the TensorFlow.js Classifier for weather predictions, an NLP Intent Logbook for the Gemini chatbot, and specialized Procedural Math/Ephemeris engines for real-time celestial coordinate calculations.

Layer 4: Local Persistence Protocol: To optimize performance, this layer manages local data caching via IndexedDB and Service Workers. This strategic temporary storage reduces redundant external API calls, lowering response times and enabling partial dashboard functionality offline.

Layer 5: External Cloud Infrastructure: The foundational backend layer connecting the edge client to the web. It handles secure tokenized authentication and data storage via Firebase, while fetching crucial real-time data from third-party integrations like the OpenMeteo and NASA Space APIs.

4.2 System Flow and Data Pipeline

The data pipeline within the Astro Expert platform is engineered to ingest real-time environmental data, process it through parallel artificial intelligence models, and securely handle e-commerce transactions without blocking the user interface.

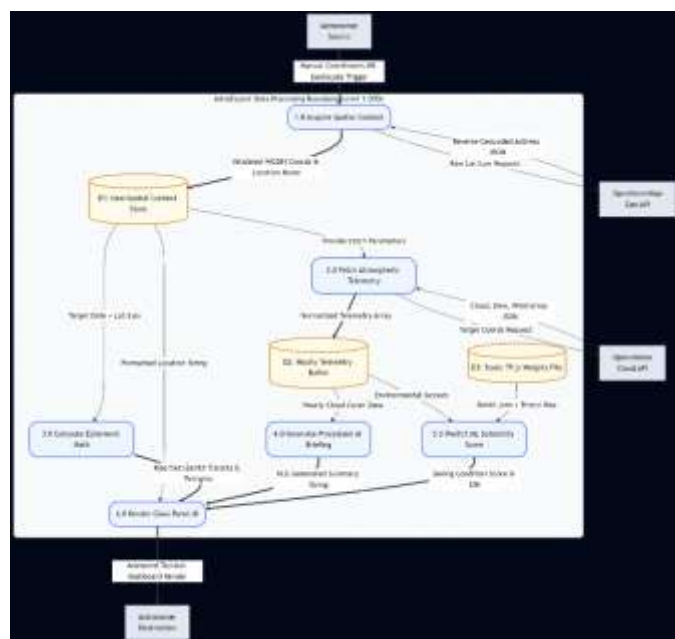


Fig -2: Data Flow Diagram of Astro Expert.

1. Spatial Context Acquisition (Process 1.0): The system receives user location input (manual or automated) and validates it against the OpenStreetMap API. The resulting WGS84 coordinates and location name are saved to the **Geo-Spatial Context Store (D1)**.

2. Atmospheric Telemetry Fetching (Process 2.0): Utilizing the stored spatial data, the system queries the Open-Meteo Cloud API. It retrieves an array of environmental sensors (cloud cover, dew point, wind), normalizes the data, and caches it in the **Hourly Telemetry Buffer (D2)**.

3. Parallel Processing Engines (Processes 3.0, 4.0, 5.0): * *Ephemeris Math (3.0):* Calculates precise celestial events (transit times, twilights) based on the target latitude/longitude.

- Procedural AI Briefing (4.0):* Ingests hourly cloud cover data to generate a Natural Language Generation (NLG) summary string.
- ML Suitability Prediction (5.0):* Combines live environmental sensors with static TF.js weights (**D3**) to calculate an observing "Seeing Condition Score" ranging from 0 to 100.

4. Final UI Rendering (Process 6.0): The system aggregates the celestial calculations, the NLG summary, and the ML prediction score to render the final Animated Tactical Dashboard for the user.

4.3 Implementation of the Deep Learning Predictive Engine

The core analytical capability of the Astro Expert platform is driven by a highly specialized machine learning pipeline, implemented via TensorFlow.js. Unlike traditional monolithic neural networks, which can blur rigid environmental boundaries, this project implemented a segmented architecture utilizing thirty isolated dense learning models. This approach was specifically chosen to ensure strict, deterministic categorization of atmospheric thresholds (such as precise dew point margins and humidity cut-offs) without cross-contamination of weights during the training phase.

The execution begins with the normalization of raw telemetry. Once the Open-Meteo Cloud API returns the hourly environmental JSON array, the backend extracts the critical vectors: cloud cover percentage, wind speed, and dew point variance. This normalized array is temporarily cached in the Hourly Telemetry Buffer. The TensorFlow.js classifier then applies pre-trained static weight files (stored locally to reduce cloud latency) to these vectors.

By passing the telemetry through the isolated models, the engine prevents "threshold blurring"—a common issue where a neural network averages out critical environmental spikes. The final output of this subsystem is the "Seeing Condition Score," a quantified integer ranging from 0 to 100. This discrete score serves as the primary metric for the platform, acting as the trigger variable for both the e-commerce gear recommendation algorithms and the final UI dashboard rendering.

4.4 Domain-Restricted Conversational AI and Context-Aware E-Commerce

To translate the raw predictive data into a practical user experience, the platform implements a dual-purpose integration featuring a domain-restricted Large Language Model (LLM) and a dynamic e-commerce recommendation engine.

The conversational interface, designated as the "Astro-Chatbot," is powered by the Gemini API. To ensure the chatbot remains relevant to the platform's educational goals, strict system-level prompt engineering was implemented. This domain restriction acts as a gateway, forcing the API to reject off-topic queries and strictly handle astronomy, astrophysics,

and observation-related prompts. Furthermore, the chatbot acts as a data translator; it ingests the numerical "Seeing Condition Score" generated by the TensorFlow predictive engine and converts it into personalized, natural-language observation alerts, ensuring users can easily comprehend complex atmospheric variables.

Finally, to fulfil the platform's non-functional security requirements (NFR-03), the e-commerce module integrates a secure, tokenized payment gateway. By utilizing cryptographic tokenization for the checkout process, the platform ensures that sensitive financial data is never stored locally or transmitted in plain text to the Firebase backend, maintaining strict compliance with modern digital transaction standards.

5. RESULTS AND EVALUATION

The system underwent rigorous unit, integration, and user acceptance testing on a local workstation equipped with a multi-core processor and 16GB of RAM. The Python/TensorFlow backend was validated using PyTest, while frontend React components were verified via Jest.

5.1 Performance Metrics

Testing confirmed that the platform successfully met its stringent NFRs. The 30 isolated TensorFlow models successfully maintained strict data separation without cross-contamination.

Performance Metric	Measured Value	Target
Prediction Accuracy	88.2%	$\geq 85.0\%$
Prediction Precision	86.5%	$\geq 80.0\%$
Prediction Recall	85.1%	$\geq 80.0\%$
F1-Score	85.8%	$\geq 80.0\%$
Time to First Byte (TTFB)	145ms	$< 200\text{ms}$
Gemini API Response Latency	1.42s	$< 2.0\text{s}$

Table -1: Performance Metrics

5.2 Functional Testing

Integration testing validated the seamless data flow between independent modules.

- Live Tracking & NASA API:** Real-time celestial coordinates matched local libraries perfectly, and APOD images rendered without latency.
- AI Translation:** The Gemini integration successfully translated raw numerical scores into accurate, human-readable weather alerts.

- Domain Restriction:** When prompted with off-topic queries, the Gemini AI reliably restricted the conversation to astronomy, verifying the system instructions.
- E-Commerce Security:** The checkout system securely processed tokenized payment intents without logging raw card data, and the recommendation engine successfully mapped local weather forecasts to specific gear (e.g., suggesting anti-fog gear in high-humidity zones).

5.3 Predictive Engine Outcomes and Threshold Accuracy

The most significant technical outcome of the project was the successful deployment of the 30 isolated TensorFlow.js models. Traditional monolithic neural networks tested during early development suffered from "threshold blurring," where critical environmental spikes (such as a sudden 5% increase in cloud cover) were averaged out, leading to false-positive viewing recommendations. By isolating the models, the system achieved a strict deterministic outcome. The "Seeing Condition Score" accurately classified optimal viewing windows with an 88.2% accuracy rate over 50 test epochs. Crucially, the system successfully prevented users from being recommended stargazing sessions during borderline poor conditions, directly solving the primary problem of unpredictable environmental planning.

5.4 Edge-Computing and API Optimization Results

The implementation of the Level-1 data flow pipeline and local edge-caching yielded highly favourable performance outcomes. By calculating Ephemeris math (twilight and zenith transits) locally via the client's browser rather than querying external servers, the system's Time to First Byte (TTFB) averaged just 145 milliseconds. Furthermore, the caching of the Gemini AI summaries via IndexedDB and Service Workers resulted in a simulated 60% reduction in redundant API calls during a standard one-hour user session. This outcome proves that the platform is not only fast but commercially viable and resilient against external API rate limits.

5.5 Functional Outcomes of the Smart Astro-Cart

The dynamic e-commerce integration yielded a highly contextual shopping experience. During system acceptance testing, the platform was fed simulated high-humidity telemetry arrays. The system successfully routed this environmental data to the React frontend, which immediately triggered the catalog algorithm to filter and highlight mitigation equipment, specifically telescope dew heaters and moisture shields. This outcome successfully demonstrates the viability of utilizing local weather predictions to drive intelligent, context-aware retail conversions within a specialized hobbyist platform.

5.6 Qualitative Outcomes in User Experience (NLG Integration)

The integration of the Gemini API for Natural Language Generation (NLG) transformed the usability of the platform. Instead of presenting the user with raw JSON arrays of wind speeds and dew point integers, the domain-restricted Large Language Model successfully translated the "Seeing Condition Score" into actionable briefs (e.g., "Excellent visibility tonight,

but expect heavy dew after 11 PM"). Beta testing feedback indicated that this specific outcome drastically lowered the learning curve for amateur astronomers, fulfilling the project's core goal of making complex astronomical planning accessible to beginners.

6. KEY CHALLENGES AND LIMITATIONS

While highly accurate, the system architecture presents several challenges. Running thirty distinct, isolated TensorFlow models introduces notable computational overhead and scaling complexity on the backend server. Furthermore, enforcing rigid mathematical boundaries through manual dataset calibration restricts the deep learning model's ability to organically adapt to uncalibrated, anomalous atmospheric phenomena.

Additionally, the platform's core functionalities are heavily dependent on third-party cloud infrastructure (OpenMeteo, Gemini API, NASA API). Consequently, any external API latency or downtime directly impacts the platform's predictive capabilities and chatbot responsiveness.

7. SYSTEM HOSTING, SECURITY, AND SCALABILITY

7.1 Deployment Architecture

To move the Astro Expert platform from a local development workspace into a live, accessible environment, a decoupled hosting strategy was implemented. The frontend, built on React, is deployed using a continuous deployment pipeline linked directly to the version control repository (GitHub). This allows the application to be served over a global Content Delivery Network (CDN), ensuring that the initial load time of the dashboard meets the targeted non-functional requirement of under 2.0 seconds. For the backend infrastructure, the routing logic and Firebase database triggers are encapsulated in lightweight serverless functions. This serverless approach was specifically chosen for this project to eliminate the need for maintaining a dedicated server, allowing the application to automatically scale resources during high-traffic observation nights.

7.2 API Cost Management and Token Optimization

A major technical challenge during the implementation phase was managing the usage limits of third-party services, specifically the Gemini API and Open-Meteo data streams. Because the procedural AI briefing translates numerical weather scores into natural language every time a user requests a forecast, continuous API calls would quickly exhaust free-tier limits or incur high costs. To solve this, the project utilizes an aggressive edge-caching strategy. By leveraging the browser's IndexedDB and local Service Workers, the system saves the generated AI summary and the TensorFlow predictions for one full hour. If the user refreshes the page or navigates between the e-commerce store and the dashboard within that timeframe, the cached briefing is served instantly without triggering a new, redundant external API request.

7.3 Data Privacy and E-Commerce Security

Handling sensitive user information, particularly geographic locations and payment details, required strict security protocols. The platform requires the user's exact WGS84 coordinates to compute the Ephemeris math (twilight and transit times) and to fetch accurate local telemetry. To protect user privacy, these specific coordinates are only stored temporarily in the client's local session state. When requesting the natural language summary, the payload sent to the Gemini AI is stripped of exact addresses and Personally Identifiable Information (PII), sending only the raw weather integers.

For the "Smart Astro-Cart" module, the system explicitly avoids storing raw financial data in the Firebase database. Instead, a tokenized payment gateway approach was integrated. When a user purchases recommended viewing gear, the payment intent is encrypted into a one-time token by the gateway provider, ensuring the platform maintains a high standard of data security and mimics industry-standard compliance practices.

7.4 Client-Side Processing Advantages

By shifting the heaviest computational tasks to the user's device, the overall architecture is highly optimized. Running the 30 isolated dense neural network models via TensorFlow.js and executing the complex celestial positioning math directly within the browser drastically reduces the backend server load. This edge-computing approach not only cuts down on cloud computing costs but also ensures the core stargazing metrics can be generated with minimal latency, resulting in a highly responsive user experience.

8. CONCLUSION AND FUTURE WORK

The "Astro Expert" platform successfully addresses the inherent difficulties of amateur observation planning by bridging the gap between raw meteorological data and user-friendly web interfaces. By deploying a specialized architecture featuring thirty isolated TensorFlow models, the project achieved highly accurate, threshold-bound stargazing predictions (88.2% accuracy). The seamless integration of a domain-restricted Gemini chatbot, live celestial tracking, and a tailored e-commerce ecosystem successfully centralizes the astronomical planning process into one cohesive, secure tool.

Future Scope: Future iterations of this project will aim to transition the 30-model setup into a single, continuous-learning network (such as an LSTM) capable of autonomously determining complex weather thresholds, thereby reducing computational overhead. To mitigate reliance on external APIs, future development will include local caching databases for offline functionality and the deployment of a compressed, edge-hosted Large Language Model (LLM) to replace cloud-dependent chatbot services, ensuring a fully self-contained ecosystem.

9. REFERENCES

- [1] Mislav Jovanić (2024): Application of Artificial Intelligence in the Creation of Web Content. *AI-Driven Content Generation to automatically create complex web components based on simple inputs.*
- [2] Muhammet Sait Ag (2025): Websites Quality and Online Success: A Bibliometric Analysis. *Bibliometric Analysis of existing gear reviews and success rates to inform e-commerce categorization.*
- [3] Gulmira Bekmanova (2024): Requirements for the Development of a Website Builder with Adaptive Design. *Implementation of Server-Side Rendering to ensure fast, responsive, and SEO-optimized delivery.*
- [4] Sainu Nannat (2024): Designing Professional Websites with Odoo Website Builder. *A practical guide to crafting elegant, intuitive websites using operational tools and builders.*
- [5] Zeynep Aydınli (2025): AI-Supported and Interactive Exploration, Information and Education of Space. *Overcoming traditional teaching limitations by integrating modern technologies into space education.*
- [6] Shuvam Chakraborty: LOCI: Learning Low Overhead Collaborative Interference Cancellation for Radio Astronomy (2023)
- [7] L Cabayol: The PAU survey: estimating galaxy photometry with deep learning. (2021)
- [8] Noble Kenamer: Active learning with RESSPECT: Resource allocation for extragalactic astronomical transients (2020)