International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

Centralized Automated Solution for Price Estimation and Reasonability

Thakur Bhargavi Walmik, Khatate Manasvi Santosh
Under the Guidance of Prof. K. R. Metha
G M Vedak Institute Of Technology
Department of Computer Engineering University of Mumbai, Maharashtra, India

Abstract- — The rapid growth of internet users has led to an increase in phishing attacks, where attackers create deceptive URLs to steal sensitive information. This study presents an ensemble machine learning framework for detecting phishing websites using Natural Language Processing (NLP) and multiple classifiers, including Artificial Neural Networks (ANN), Naïve Bayes (NB), Random Forest (RF), and Support Vector Machines (SVM). By extracting key features from URLs and applying machine learning techniques, the proposed model enhances detection accuracy. Comparative analysis demonstrates its effectiveness, achieving 98.4% accuracy in distinguishing phishing sites from legitimate ones. This approach offers a proactive solution to mitigate online security threats and protect users from cyber fraud. Phishing attacks have become more sophisticated, using deceptive URLs to target unsuspecting users. This research introduces a hybrid machine learning-based detection model that enhances accuracy through an ensemble of classifiers. The system utilizes Natural Language Processing (NLP) to extract critical URL features, which are then analyzed using Artificial Neural Networks (ANN), Naïve Bayes (NB), Random Forest (RF), and Support Vector Machines (SVM). Machine learning techniques are particularly effective in detecting zero-hour phishing attacks and adapting to emerging threats. Our implementation achieved a 98.4% accuracy in classifying websites as phishing or legitimate.

Keywords: Centralized price estimation, automated procurement system, price reasonability analysis, web scraping, Google Shopping integration, Node.js, Express.js, Puppeteer, headless browser automation, real-time price extraction, user authentication, dynamic profile management, responsive web application, tender management system, data extraction efficiency, procurement decision support, LocalStorage session management, multi-platform scalability, price comparison system, ERP integration readiness.

I. INTRODUCTION

1.1 Background

The project "Centralized Automated Solution for Price Estimation and Reasonability" aims to make price comparison easier and faster for users. It focuses on collecting and analyzing prices of Smartphones and their different model from multiple online platforms. The system automatically gathers price details, compares them, and shows the most reasonable price in one place. This helps users save time, avoid confusion, and make smarter buying decisions.

The Centralized Automated Solution for Price Estimation Reasonability is a web-based platform designed to address this challenge. This project provides an automated mechanism to compare product prices across major Indian ecommerce platforms — Amazon India, Flipkart, and Croma — enabling users to make informed purchasing decisions.

1.2 Problem Statement

In today's online market buyers have to face the following challenges when estimating reasonable prices for products:

- Price Disparity: The same product is listed at different prices on different e-commerce platforms, making it difficult to determine the true market value.
- Manual Effort: Comparing prices manually across Amazon, Flipkart, Croma, and other platforms is extremely time-consuming and error-prone.
- Lack of Centralization: There is no single unified platform that aggregates prices from multiple trusted Indian e-commerce portals for procurement-related price estimation.
- Dynamic Pricing: E-commerce platforms frequently change prices based on demand, offers, and seasonal discounts, making static price references unreliable.
- Transparency: Tender processes require transparent and verifiable price references, which are hard to obtain without automation.

1.3 Objective

The primary objectives of this project are:

- To develop a centralized web-based platform that compares product prices across three major Indian e-commerce websites: Amazon India, Flipkart, and Croma.
- To support multi-brand smartphone price comparison covering five major brands: Apple, Samsung, Oppo, Realme, and Vivo with a database of 50 products.
- To implement web scraping functionality using Puppeteer for real-time price extraction from Google Shopping.
- To provide direct navigation links to e- commerce platforms for verifying real-time prices and offers. To create an intuitive, responsive user interface with brand-wise filtering, product cards, and comparison tables.
- To build a scalable architecture that can be extended to additional product categories and ecommerce platforms.

1.4 Scope of the Project

The scope of this project encompasses the development of a full-stack web application that serves as a price estimation and comparison tool. The system covers:

1. Product Coverage: 50 smartphone models across 5 brands (Apple, Samsung, Oppo, Realme, Vivo) with 10 models per brand.
2. E-Commerce Platforms: Price comparison across Amazon India, Flipkart, and Croma.
3. User Management: Registration, login, and profile management functionality.
4. Tender Management: Interface for publishing and viewing tenders related to procurement.
5. Web Scraping: Google Shopping scraper using Puppeteer for fetching real-time product data.

II. LITERATURE REVIEW

2.1 Primary Research

Title: Price Comparison Application for E- Commerce Using Web Scraping Authors: Dhulipalla Tejaswi, Karumanchi Nikitha, Munji Mounika, Dammavalam Sai Kamakshi Harshitha, Kunchala Sirisha. Published In: International Advanced Research Journal in Science, Engineering and Technology (IARJSET), Vol. 12, Issue 4, April 2025

Overview:
This paper proposes an automated system for price comparison across e-commerce platforms using web scraping. The system collects real- time price data from websites like Amazon, Flipkart, Myntra, Croma, Nykaa, OLX, and Google Shopping. It identifies matching products based on model and specifications, compares prices, and highlights the best deal. Developed using Python, Flask, and Streamlit with an SQLite database, the system also provides user features such as authentication, wishlist management, and search tracking. The results showed over 91 % accuracy in product matching and reduced manual effort in finding the best price.

This research supports the concept of a centralized, automated price estimation system that improves transparency, efficiency, and decision-making for online shoppers.

Title: Development of an Automated Tool for Cost Estimation of Transportation Projects Authors: Behzad Rouhanizadeh, Sharareh Kermanshachi, Issa J. Ramaji, Shahrad Shakerian.
Published In: American Society of Civil Engineers (ASCE), 2020

Overview:
This paper presents a BIM-based automated system for improving the accuracy and efficiency of cost estimation in transportation projects. The tool integrates Industry Foundation Classes (IFC) and Level of Development (LOD) concepts within a web-based framework built using C# and ASP.NET MVC Core. It automates quantity take- off, updates costs dynamically, and minimizes errors.
The results show that increasing the LOD enhances cost accuracy and reliability, helping project managers make more informed financial decisions. The study demonstrates the benefits of automation and standardization for faster and more transparent cost estimation.

2.2 Secondary Research

Existing platforms like Google Shopping, PriceDekho, and Smartprix were studied to understand their price comparison methods. Most tools display prices but lack automation and detailed price analysis. This research helped identify gaps to develop a more accurate and centralized price estimation system.

2.3 Comparative Analysis of Existing System PriceDekho

PriceDekho compares product prices from sites like Amazon and Flipkart. It shows basic details but doesn't provide real-time updates or analyze price accuracy.
Smartprix

Smart price compares prices and features of products through a clean interface. However, it lacks automation and doesn't justify price differences or estimate reasonable prices.

My Smart Price

MySmartPrice lists product prices and offers from multiple e-commerce sites. It provides periodic updates but no real-time data or price reasonability analysis.

2.4 Research Gap Analysis

Existing platforms like Price Dekho, Smart price and MySmartPrice only display product prices without real-time updates or accuracy checks. They lack automation and do not analyse price reasonability. There is a gap in developing a centralized automated system that can estimate and compare real-time prices accurately for products like smartphones.

III. SYSTEM ANALYSIS AND DESIGN

3.1 System Architecture

The system follows a Model-View-Controller (MVC) architectural pattern with a Node.js/Express backend. The architecture consists of three primary layers:

1. Presentation Layer (Views): EJS (Embedded JavaScript) templates render dynamic HTML pages. The frontend uses Bootstrap 4 for responsive design and Font Awesome for iconography. The homepage (index.html) uses vanilla JavaScript with the Fetch API to dynamically load product data from the backend REST API.
2. Application Layer (Controllers): The Express.js server (app.js) handles routing, request processing, and business logic. It exposes REST API endpoints for fetching smartphone data and manages multiple query workflows including brand-based comparison, Google Shopping scraping, and specification-based search.
3. Data Layer (Models): The smartphone database (smartphone-database.js) serves as the primary data store, containing 50 product records with MRP prices and dynamically generated search URLs for three ecommerce platforms. The URL generation functions use JavaScript's encodeURIComponent() to create properly encoded search queries.

3.2 Technology stack

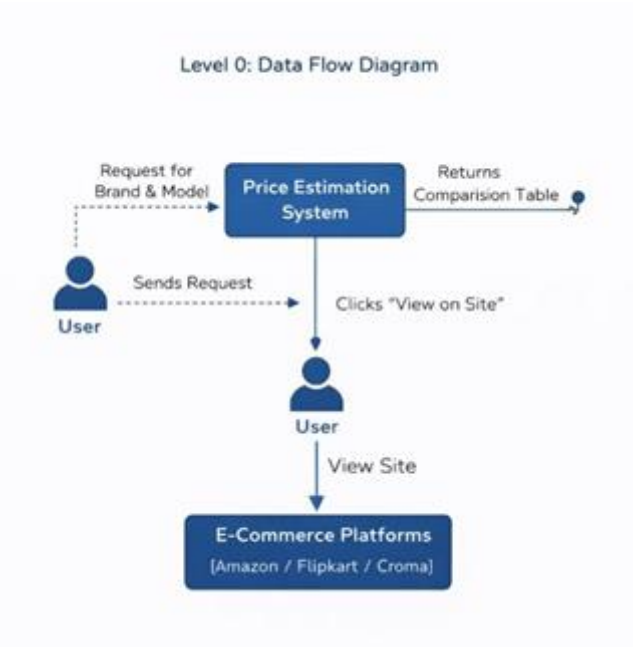
Component	Technology	Version	Purpose
Runtime Env	Node.js	18.x+	Server-side JavaScript execution
Web Framework	Express.js	4.21.2	HTTP server, routing, middleware
Template Engine	EJS	3.1.10	Server-side HTML rendering
Web Scraper	Puppeteer	23.11.1	Headless Chrome browser automation

Frontend Framework	Bootstrap	4.5.0	Responsive CSS framework
Icons	Font Awesome	6.0.0	Scalable vector icons
Client-Side Library	jQuery	3.5.1	DOM manipulation, AJAX requests

3.3 Data flow Diagram

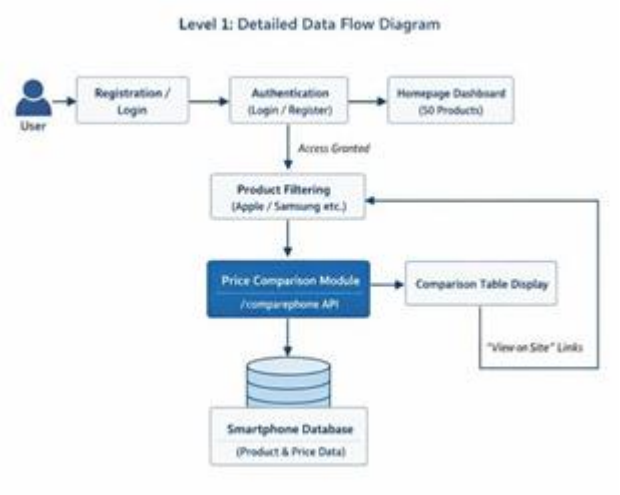
The data flow through the system follows these steps:

Level 0 (Context Diagram):



Level 1 (Detailed Flow):

- User accesses the Registration Page and logs in with credentials.
- System redirects to the Homepage Dashboard displaying 50 smartphone product cards fetched via the
- /api/smartphones REST API.
- User selects a brand filter (Apple / Samsung
- / Oppo / Realme / Vivo) to narrow down products.
- User clicks "Compare Prices" on a product card, triggering a GET request to
- /comparephone?brand=X&model=Y.
- The server queries the smartphone database, retrieves price data for 3 e-commerce platforms, and renders the comparison view.
- The comparison table displays MRP prices and provides "View on Site" links that perform a search query on each e-commerce platform to show real-time prices.



IV. DATABASE DESIGN

The smartphone database is implemented as JavaScript module rather than a traditional relational database, making it lightweight and suitable for the application's requirements. The database structure is as follows:

- name (String): Stores the full product name (e.g., Apple iPhone 16 Pro Max 256GB).
- price (String): Contains the formatted MRP price along with the currency symbol (e.g., ₹1,44,900 (MRP)).
- priceValue (Number): Stores the numeric value of the price used for sorting and comparison (e.g., 144900).
- link (String): Holds the dynamically generated search URL for the respective ecommerce platform.
- source (String): Indicates the e- commerce platform name (Amazon India
- / Flipkart / Croma).
- available (Boolean): Represents the availability status of the product (True/False).

- brand (String): Specifies the smartphone brand name (Apple / Samsung / Oppo / Realme / Vivo).
- image (String): Stores the file path of the product image used for display.

V. IMPLEMENTATION

5.1. Module Description

The application is organized into several distinct modules, each responsible for a specific aspect of the system's functionality. The modular design ensures separation of concerns, maintainability, and ease of future enhancements.

5.2. Core Modules

5.2.1 Express Server (app.js)

The main application server is built using Express.js and handles all HTTP routing, middleware configuration, and view rendering. It listens on port 3000 (configurable via environment variable) and serves static files from both the project root and the web_scraper_engine_v1 directory. Key routes include:

- GET / — Serves the registration page as the entry point.
- GET /home — Renders the Phone Query dashboard (phone_query.ejs).
- GET /api/smartphones — REST API
- endpoint returning all 50 products and brand configurations as JSON.
- GET /compare-phone — Accepts brand and
- model parameters, queries the database, and renders the comparison view.
- GET /search — Handles specification-based product searches using the Puppeteer web scraper.
- GET /compare-iphone — Legacy route for iPhone-specific comparison with real-time scraping fallback.

parameters, reducing code duplication. The database also exports brand configuration data including display names, icons, and theme colors for each of the five brands.

5.2.3 Web Scraper Module (scraper.js)

The web scraper module utilizes Puppeteer to automate a headless Chrome browser for extracting product data from Google Shopping. When a user submits a specification- based search query, the scraper constructs a Google Shopping URL, navigates to the page, waits for the content to load, and then uses DOM selectors to extract product titles, prices, images, source retailers, and product URLs.

The scraper is configured with sandbox disabling flags for compatibility and implements error handling with graceful fallback to return empty results.

5.2.4 View Templates (EJS Files)

The presentation layer consists of seven EJS templates:

iphone_results.ejs	Legacy iPhone search results display

5.2.2 Smartphone Database Module (smartphonedatabase.js)

This module serves as the central data repository for the application. It contains 50 smartphone products organized by model name, with each product having three entries corresponding to Amazon India, Flipkart, and Croma. The module uses a helper function createProductEntries() that generates all three platform entries from a single set of product

5.2.5 Static Frontend Pages

The application includes several static HTML pages:

- registration.html: User login/registration page with a dark-themed UI, form validation, and localStorage-based session management.
- index.html: Main dashboard with a dynamic product grid that fetches data from the/api/smartphones API, brand filter tabs, and product cards with "Compare Prices" buttons.
- profile-settings.html: User profile management page for updating personal information.
- tender/publish_tender.html: Interface for creating and publishing procurement tenders.
- tender/view_tender.html: Interface for viewing published tenders.

VI. FEATURES AND USER INTERFACE

6.1 User Registration and Authentication

The application starts with a registration/login page that features a modern dark-themed design with green accent colors. User authentication is implemented using client- side localStorage, providing a lightweight session management mechanism. The registration page includes fields for name, email, mobile number, and password. Upon successful login,

the user's credentials are stored locally and their profile information is accessible throughout the application via a profile popup accessible from the navigation bar.

The profile management section allows users to view and update their personal information. A profile icon in the navigation bar

Template File	Purpose
phone_query.ejs	Brand and model selection form with dynamic dropdown menus for all 5 brands
phone_comparison.ejs	Price comparison table showing MRP across 3 e-commerce platforms with sorting and highlighting
query_1.ejs	Specification-based product search form with title, specs, make, and model fields
query_2.ejs	Basic requirements-based product search with title and specs fields
iphone_query_1.ejs	Legacy iPhone-specific search interface
iphone_comparison.ejs	Legacy iPhone-specific comparison view with real-time price indicators

provides quick access to user details, while a dedicated Profile Settings page enables comprehensive profile editing.

6.2 Product Dashboard

The main dashboard (index.html) serves as the central hub of the application. It dynamically fetches all 50 smartphone products from the backend API and renders them as a responsive card grid. Key features of the dashboard include:

- Brand Filter Tabs: Five brand-specific filter buttons (All, Apple, Samsung, Oppo, Realme, Vivo) allow users to quickly narrow down products by manufacturer. The active filter is visually highlighted with the application's primary color (#667eea).
- Product Cards: Each product is displayed in a card format showing the product image, brand badge with brand-specific color coding, model name, MRP price, and a "Compare Prices" action button.
- Responsive Grid Layout: The product grid uses CSS Grid with auto-fill and minmax(220px, 1fr) to automatically adjust the number of columns based on screen size, ensuring optimal display on all devices.
- Loading Animation: When a user initiates a price comparison, a loading overlay with a
- spinner provides visual feedback while the comparison page loads.

6.3 Price Comparison View

The price comparison view (phone_comparison.ejs) provides a detailed sideby-side comparison of a selected product's pricing across three e-commerce platforms. The view includes:

- Product Header: Displays the product image, brand badge with brand-specific color, and the full product name.
- Information Notice: An informational banner explains that displayed prices are MRP references and encourages users to click "View on Site" for real-time prices and offers.
- Comparison Table: A styled table with gradient header showing Product Name, Price, Brand, Source (with color-coded badges for Amazon, Flipkart, and Croma), and an Action column.
- Best Price Highlighting: The lowest-priced entry is automatically highlighted with a green background and a "Best Price!" badge. Products are sorted by price in ascending order.
- View on Site Links: Each entry has a "View on Site" button that opens a search query on the respective e-commerce platform in a new tab, displaying the product's current listing with realtime pricing.
- Availability Indicators: Out-of-stock products are clearly marked with red styling and a disabled action button.

6.4 Query System

The application supports query mode, accessible from the homepage popup menu:

Query Mode — Smartphone Price Comparison: The primary query mode allows users to select a brand and model from dropdown menus to compare MRP prices across Amazon India, Flipkart, and Croma. The model dropdown dynamically populates based on the selected brand, showing all 10 models available for each brand.

VII. TESTING AND RESULTS

7.1 Testing Methodology

The application was tested using a combination of manual testing, functional testing, and integration testing approaches. Testing was conducted on Windows operating systems using Google Chrome, Microsoft Edge, and Mozilla Firefox browsers. The Node.js server was tested for stability under continuous operation and for handling concurrent user requests.

7.2. Test cases

- TC-01: User Registration o Description: User Registration o Input: Valid name, email, mobile, password o Expected Output: User created, redirected to login
- Status: Pass
- TC-02: User Login o Description: User Login o Input: Valid email and password o Expected Output: Redirected to dashboard o Status: Pass
- TC-03: Dashboard Product Loading o Description: Dashboard Product Loading o Input: Open dashboard page o Expected Output: 50 product cards displayed with brand filters
- Status: Pass
- TC-04: Brand Filter - Apple o Description: Brand Filter - Apple o Input: Click "Apple" filter button
- Expected Output: Only 10 Apple products displayed
- Status: Pass
- TC-05: Price Comparison o Description: Price Comparison o Input: Select Samsung Galaxy S24 Ultra o Expected Output: Comparison table with 3 entries (Amazon, Flipkart, Croma)
- Status: Pass
- TC-06: View on Site - Amazon o Description: View on Site - Amazon
- Input: Click "View on Site" for Amazon link o Expected Output: Opens Amazon India search page for the product
- Status: Pass
- TC-07: View on Site - Flipkart o Description: View on Site - Flipkart o Input: Click "View on Site" for Flipkart link o Expected Output:
- Opens Flipkart search page for the product
- Status: Pass
- TC-08: View on Site - Croma o Description: View on Site - Croma o Input: Click "View on Site" for Croma link o Expected Output:
- Opens Croma search page for the product
- Status: Pass

VIII. CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

The project “Centralized Automated Solution for Price Estimation and Reasonability” successfully provides an automated and efficient method for comparing iPhone prices across multiple e-commerce platforms. It simplifies the buying process by collecting real-time data, analyzing it, and displaying the most reasonable price on a single platform. This system helps users save time, avoid confusion, and make smart purchasing decisions.

Through the use of Node.js, Express.js, EJS, and Puppeteer, the project ensures automation, accuracy, and ease of use. The interface is simple, responsive, and user-friendly, making it accessible to all types of users.

The key achievements of this project include:

- Successfully developed a full-stack web application using Node.js, Express.js, and EJS that compares smartphone prices across Amazon India, Flipkart, and Croma.
- Built a comprehensive database of 50 smartphone models spanning 5 major brands, with MRP pricing and dynamically generated search URLs that reliably navigate users to the correct product listings.
- Implemented a Google Shopping web scraper using Puppeteer for real-time, specification- based product searches beyond the smartphone database.

- Created an intuitive and responsive user interface with brand filtering, product cards, comparison tables, best-price highlighting, and direct ecommerce platform links.
- Developed a modular and extensible architecture that supports multiple query modes and can be expanded to additional product categories and platforms.
- Integrated user management, profile settings, and tender management features to support procurement workflows.

8.2 Future Scope

The following enhancements are planned or recommended for future iterations of the project:

1. Database Integration: Migration from JavaScript-based flat-file database to MongoDB or PostgreSQL for better scalability, querying, and data persistence.
2. Real-Time Price Updates: Integration with official e-commerce APIs (Amazon Product Advertising API, Flipkart Affiliate API) for fetching live prices without scraping.
3. Product Category Expansion: Extending the platform beyond smartphones to cover laptops, electronics, office equipment, networking hardware, and other categories relevant to government procurement.
4. Price History and Trends: Implementing price tracking over time with historical charts to identify best purchasing windows and price trends.
5. Email Notifications: Price drop alerts and periodic comparison reports sent via email to registered users.
6. AI-Based Price Prediction: Training ML models on historical pricing data to predict future price movements and estimate fair market values.
7. Multi-Language Support: Adding support for Hindi and other regional languages to improve accessibility for government officials across India.
8. Mobile Application: Developing a companion mobile application (React Native or Flutter) for on-the-go price comparisons during field procurement.
9. User Authentication Enhancement: Implementing server-side authentication with JWT tokens and bcrypt password hashing for improved security over localStorage-based sessions.

REFERENCES

1. Mitchell, R. (2018). Web Scraping with Python: Collecting Data from the Modern Web. O'Reilly Media, 2nd Edition.
2. Google Chrome Labs (2024). Puppeteer Documentation. Available at: https://pptr.dev/
3. Node.js Foundation (2024). Node.js Official Documentation. Available at: https://nodejs.org/en/docs/
4. Express.js Team (2024). Express.js Framework Guide. Available at: https://expressjs.com/
5. Mozilla Developer Network (2024). JavaScript and Web APIs Reference. Available at:
6. https://developer.mozilla.org/
   Mozilla Developer Network (2024). JavaScript encodeURIComponent() Reference. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/encodeURIComponent
7. Bootstrap Team (2020). Bootstrap 4.5 Documentation. Available at: https://getbootstrap.com/docs/4.5/
8. Chart.js Team (2024). Chart.js Documentation. Available at: https://www.chartjs.org/docs/
9. EJS Team (2024). EJS (Embedded JavaScript Templates) Documentation. Available at: https://ejs.co/
10. Font Awesome Team (2024). Font Awesome Icons Documentation. Available at: https://fontawesome.com/
11. jQuery Team (2024). jQuery Documentation. Available at: https://jquery.com/
12. Kumar, S. & Rahman, Z. (2015). "Electronic Procurement Systems: A Review." International Journal of Procurement Management, 8(1/2), pp. 134163.
13. Zhao, Y., Chen, L. & Wang, J. (2020). "Automated Data Collection from Dynamic Websites." Journal of Web Engineering, 19(3), pp. 245-268.
14. Khder, M. A. (2021). "Web Scraping: State of Art, Techniques and Applications." International Journal of Advances in Soft Computing, 13(3), pp. 144-168.
15. Glez-Peña, D. et al. (2014). "Web Scraping Technologies in an API World." Briefings in Bioinformatics, 15(5), pp. 788-797.
16. Amazon India (2024). Official Website. Available at: https://www.amazon.in/
17. Flipkart (2024). Official Website. Available at: https://www.flipkart.com/
18. Croma (2024). Official Website. Available at: https://www.croma.com/
19. Google Shopping (2024). Google Shopping Platform. Available at: https://shopping.google.com/
20. Smart India Hackathon (SIH) (2024). Official Website. Available at: https://www.sih.gov.in/

21. National Technical Research Organisation (NTRO) (2024). Government of India. Available at: Government of India Official Sources.