**Contents**

1. **Introduction ................................................................................................................. 1  
   1.1 Abstract ..................................................................................................................... 1  
   1.2 Purpose ...................................................................................................................... 1**

**1.3 Scope .......................................................................................................................... 1  
1.4 Overview .................................................................................................................... 2**

1. **Overall Description .................................................................................................... 3  
   2.1 Project Perspective ................................................................................................... 3  
   2.2 Project Functions ...................................................................................................... 3  
   2.3 User Classes and Characteristics ............................................................................. 4**
2. **Existing System Drawbacks & Proposed Improvements ....................................... 4  
   3.1 Overview of Existing/Traditional Systems .............................................................. 4  
   3.2 Identified Drawbacks in Existing Systems ........................................................... 4  
   3.3 Justification for Proposed System ........................................................................... 4**
3. **High-Level Architecture & System Design ..................................................... 5  
   4.1 System Overview .................................................................................................... 5  
   4.2 System Architecture Diagram .................................................................................. 6**
4. **Specific Requirements ................................................................................................ 6  
   5.1 Functional Requirements (FRs) .............................................................................. 6  
   5.2 Non-Functional Requirements (NFRs) ................................................................. 7  
   5.3 System Interfaces ...................................................................................................... 7**
5. **Appendices ................................................................................................................... 8  
   6.1 Glossary ..................................................................................................................... 8  
   6.2 References ................................................................................................................. 9**

**1. Motivation**

**1.1 Limitations of Existing Systems**

**Despite the widespread use of smart device recommendation platforms and online marketplaces, the current generation of solutions remains limited in scope and functionality. Most existing systems are characterized by the following key drawbacks:**

|  |  |
| --- | --- |
| Drawback / Limitation | Impact on Users or Marketplaces |
| Static Recommendation Engines | **Users receive generic, non-personalized suggestions.** |
| No Explainable AI (XAI) | **Users cannot understand why a particular device is recommended, reducing trust and transparency.** |
| Limited Real-Time Data Processing | **Device specifications, prices, and reviews become outdated quickly, leading to inaccurate comparisons.** |
| No Persona or Community-Based Matching | **Users lack access to community-driven trends or personalized “people like you” insights.** |
| No Gamification or Engagement Mechanisms | **User retention is low, and there is minimal incentive for repeat visits or active participation.** |

**These limitations result in a less effective, non-transparent, and less engaging user experience, which does not fully support informed decision-making or foster community involvement.**

**1.2 Project Objectives**

**The objective of the CANSIS (Cloud-AI Native Smartphone Intelligence Software) project is to address these gaps by delivering a unified, intelligent, and user-centric platform. The main goals are:**

|  |  |
| --- | --- |
| Existing Drawback | Our Proposed Solution |
| Static Recommendations | **Integrate an AI Persona Engine and collaborative filtering to provide highly personalized device suggestions.** |
| No Explainable AI | **Implement Explainable AI (XAI) methods to offer clear, understandable justifications for each recommendation (“Why this phone?”).** |
| Outdated Information | **Utilize real-time web scraping and automated database updates (Firebase/MongoDB) to ensure users always have access to the latest specs, prices, and reviews.** |
| No Persona or Community Matching | **Enable Community AI Matchmaking, displaying insights like “Users like you bought X” to build trust and confidence.** |
| Poor Scalability | **Employ a cloud-native, microservices-based architecture with optional Kubernetes auto-scaling for robust performance and reliability.** |
| No Gamification or Engagement | **Introduce leaderboards, polls, badges, and community features to enhance user engagement and retention.** |

**3. Assumptions and Constraints**

**3.1 Assumptions**

The design and implementation of the CANSIS platform are based on the following assumptions:

1. **Data Availability and Access**
   * Reliable and structured data can be obtained from public sources such as GSMArena, Flipkart, Amazon, and other online marketplaces.
   * APIs or web scraping will continue to be feasible for extracting device specifications, prices, and reviews.
2. **User Base Characteristics**
   * Target users will have access to modern web browsers and stable internet connectivity.
   * Users may span a variety of languages and accessibility needs, motivating multilingual and voice-based interaction features.
   * A portion of users will value explainability, community input, and gamification for engagement.
3. **Technology Stack and Integration**
   * Chosen cloud services (Firebase, MongoDB Atlas, Railway, Render, etc.) will remain available and provide necessary free tiers for initial deployment.
   * Third-party APIs (Hugging Face, Whisper, Google Speech-to-Text, etc.) will allow reasonable usage within their free or trial tiers.
   * Open-source machine learning frameworks (scikit-learn, XGBoost, LSTM, spaCy, etc.) will continue to be maintained and supported.
4. **Security and Privacy**
   * User authentication (Firebase Auth, Google reCAPTCHA) will provide adequate baseline protection against unauthorized access and spam.
   * Users consent to basic data collection for improving recommendations and engagement features, compliant with standard privacy norms.
5. **Team and Resource Availability**
   * The core development team will be capable of full-stack development (backend, ML, frontend, DevOps) as per the modular breakdown.
   * Sufficient resources (compute, storage, cloud credits) will be available for prototyping and early production deployment.

**3.2 Constraints**

The CANSIS project is subject to the following constraints, which may affect the scope, architecture, or deployment:

1. **Legal and Ethical Constraints**
   * All web scraping and data collection will respect website terms of service and copyright laws.
   * User privacy regulations (e.g., GDPR, local data protection acts) must be followed for data storage, processing, and user consent.
2. **Technical Constraints**
   * Resource limitations on free cloud tiers (Firebase, MongoDB Atlas, Railway, etc.) may restrict the scale of storage, bandwidth, and compute resources, especially during rapid growth or high traffic.
   * Rate limits and usage quotas on third-party APIs may impact the frequency and scope of features such as sentiment analysis, voice recognition, and model hosting.
   * The initial deployment is focused on web-based access; mobile app versions are out of scope for the MVP.
   * Real-time data scraping and updates depend on the stability and structure of third-party websites, which may change unexpectedly.
3. **Performance Constraints**
   * Real-time recommendation and personalization features require efficient ML inference and database access within acceptable latency limits.
   * Scalability for high user loads is limited by available infrastructure and may require additional resources or optimization for enterprise deployment.
4. **Security Constraints**
   * Sensitive operations (user authentication, leaderboard data, rewards, etc.) must be securely handled to prevent abuse, data breaches, or manipulation.
   * API keys, database credentials, and other secrets must be protected using best practices (ConfigMaps/Secrets in Kubernetes, environment variables, etc.).
5. **Development and Maintenance Constraints**
   * The project schedule is limited to an 8-week development window, which may require strict prioritization and phased delivery.
   * Continuous updates to device databases, AI models, and engagement features will require maintenance resources beyond initial deployment.

**5. Methodology**

The methodology for the CANSIS project follows a modular, agile, and cloud-native approach to ensure rapid development, flexibility, and scalability. The process integrates best practices from software engineering, machine learning, and DevOps to deliver a reliable, maintainable, and user-centric platform.

**5.1 Modular System Architecture**

The system is organized into four main layers and corresponding modules:

* **AI & ML Intelligence Layer**: Responsible for core intelligence such as persona matching, performance simulation, spec forecasting, and explainable recommendations.
* **Marketplace & Data Intelligence Layer**: Manages real-time scraping, data collection, price/stock alerting, and historical trend storage.
* **User Interaction & Engagement Layer**: Handles all user-facing features, including the frontend UI, gamification, community modules, and voice interaction.
* **Cloud & DevOps Infrastructure Layer**: Ensures reliable hosting, scaling, automation, monitoring, security, and deployment.

Each layer is developed as an independent, loosely-coupled module, facilitating team parallelism, easier maintenance, and optional migration to microservices.

**5.2 Agile & Iterative Development**

* The project is divided into short development sprints (typically 1–2 weeks).
* Each sprint delivers incremental functionality, starting from the core backend and frontend, and progressing to advanced features like AI lab, live voice, and Kubernetes deployment.
* Regular sprint reviews and retrospectives are held to adapt to feedback and adjust priorities.

**5.3 Technology Stack & Tools**

* **Backend**: Python (Flask or FastAPI), with RESTful API endpoints.
* **Frontend**: React.js or Flask-HTML/CSS/JS for modular, responsive UI.
* **Database**: Firebase or MongoDB Atlas for real-time and historical data.
* **Machine Learning**: scikit-learn, XGBoost, LSTM, Hugging Face Transformers, spaCy.
* **Scraping & Automation**: Scrapy, BeautifulSoup, Railway/Cloud Scheduler for automated tasks.
* **Cloud Hosting**: Render, Railway, Vercel for backend/frontend; Hugging Face Spaces or AWS Lambda for model hosting.
* **Monitoring & Alerts**: Prometheus, Grafana, UptimeRobot, Twilio, SendGrid, FCM.
* **Containerization/Orchestration**: Docker, Docker Compose, optional Kubernetes (Minikube/GKE).

**5.4 Data Flow & Processing**

1. **Data Ingestion**: Automated scrapers and APIs gather device specs, prices, and reviews, which are cleaned and stored in cloud databases.
2. **AI/ML Processing**: User queries and profile data are processed by the ML engine for persona clustering, performance prediction, and spec forecasting.
3. **Recommendation & Personalization**: The backend combines AI outputs, explainable reasoning, and community trends to generate tailored device suggestions.
4. **User Interaction**: The frontend provides modular tools—comparison UI, modular need builder, live voice query, and gamification—for a seamless user experience.
5. **Monitoring & Automation**: Cloud schedulers handle periodic updates, alerts are sent on price drops/stock changes, and monitoring tools track uptime and performance.

**5.5 Security & Compliance**

* **User Authentication**: Handled via Firebase Auth and Google reCAPTCHA.
* **Data Protection**: All sensitive data is stored securely, with secrets managed via environment variables or Kubernetes secrets.

**5.6 Quality Assurance**

* Unit and integration tests are implemented for backend, ML models, and critical frontend flows.
* CI/CD pipelines (GitHub Actions, Railway/Render hooks) automate builds, testing, and deployment.
* Performance and load testing are conducted to ensure scalability and responsiveness.

**5.7 Deployment & Monitoring**

After the complete build and testing, the project is deployed using GitHub for version control, Railway for application hosting, and automated CI/CD pipelines to streamline integration, testing, and delivery. This ensures each release is robust, reliable, and can be quickly updated as needed.

**4. High-Level Architecture**

**4.1 System Overview**

The **CANSIS** platform is organized into three primary modules, each designed for modular development and optional scaling via cloud-native microservices:

**Module 1 – Data & Marketplace Intelligence**

* Scrapes device specifications from multiple sources (e.g., GSMArena, Flipkart, Amazon).
* Stores and maintains historical data in Firebase or MongoDB.
* Sentiment Analysis on User Reviews.

**Module 2 – AI & ML Intelligence**

* Consumes cleaned marketplace and device data from Module 1.
* Generates:
  + AI persona-based device recommendations
  + Device performance predictions (AI Lab)
  + Device comparisons
  + Forecasts of upcoming device specifications
  + Explainable AI insights for transparency
* Returns predictions and insights to Module 3 for user-facing visualization.

**Module 3 – User Interaction & Cloud Deployment**

* Provides the main web interface and optional mobile interface.
* Displays AI-driven recommendations and gamification features for user engagement.
* Handles voice-based queries and user account management.
* Supports flexible cloud deployment via platforms such as Render, Railway, or optional Docker/Kubernetes microservices for enterprise scalability.

**4.2 System Architecture Diagram**

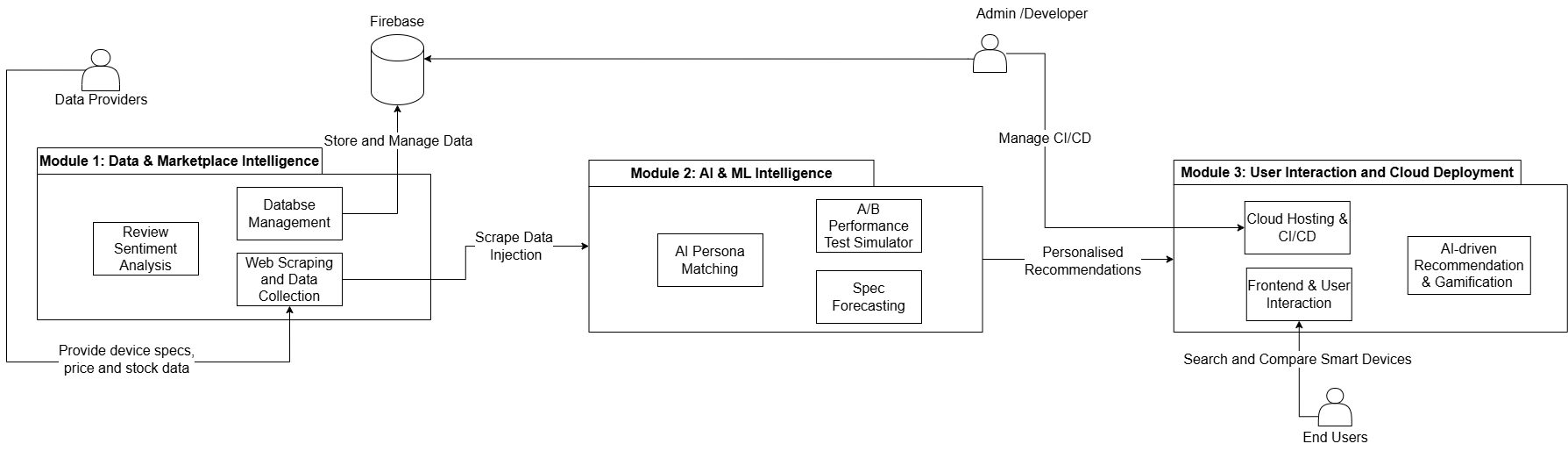


Fig. 4.1 High-Level Architecture Diagram

**5. Specific Requirements**

**5.1 Functional Requirements (FRs)**

The system shall provide the following key functionalities:

|  |  |
| --- | --- |
| FR ID | Functional Requirement |
| FR-1 | The system shall scrape real-time device specifications from multiple online sources (e.g., GSMArena, Flipkart, Amazon). |
| FR-2 | The system shall store cleaned, structured data in a cloud database (Firebase or MongoDB Atlas) for efficient real-time access and processing. |
| FR-3 | The system shall ensure all data scraping and updates occur automatically through scheduled tasks (cron jobs or cloud schedulers). |
| FR-4 | The system shall generate personalized smartphone recommendations based on user personas using AI/ML clustering techniques. |
| FR-5 | The system shall predict device performance using machine learning models (e.g., XGBoost, regression) within an AI Lab feature. |
| FR-6 | The system shall forecast future device specifications using time-series models and large language models (LLMs). |
| FR-7 | The system shall provide explainable AI (XAI) insights to offer transparent reasoning behind recommendations. |
| FR-8 | The system shall offer a web-based user interface for device comparison, timeline visualization, and interactive gamification features. |
| FR-9 | The system shall display leaderboards, polls, badges, and similar engagement mechanisms to enhance user participation. |
| FR-10 | The system shall support scalable backend hosting on platforms such as Render or Railway, with optional deployment as microservices via Kubernetes. |

**5.2 Non-Functional Requirements (NFRs)**

|  |  |
| --- | --- |
| Category | Non-Functional Requirement |
| Performance | The system shall process and update device data **within 10 minutes of scraping**. |
| Scalability | The system shall support **auto-scaling using Kubernetes HPA** to handle traffic spikes. |
| Availability | The system shall ensure **99% uptime** using cloud-native hosting with monitoring. |
| Security | All data transfers shall use **HTTPS & Firebase Auth** with **Google reCAPTCHA**. |
| Usability | The system shall provide a **responsive, mobile-friendly web interface**. |
| Maintainability | The system shall use **modular microservices** to allow easy updates and maintenance. |
| Portability | The backend shall run on **Docker containers** to allow **cloud and local deployment**. |

**5.3 System Interfaces**

**5.3.1 User Interfaces**

* **Web Dashboard:**
  + Displays device comparison, gamification leaderboards, and AI persona recommendations.
  + Provides **voice query** functionality for hands-free search.

**5.3.2 Hardware Interfaces**

* **End-User Devices:**
  + Minimum: Dual-core CPU, 4GB RAM, modern browser
* **Server Environment:**
  + Cloud-hosted Docker containers (1 vCPU, 1–2GB RAM per service)
  + Optional Kubernetes pods for scaling

**5.3.3 Software Interfaces**

* **Databases:** Firebase / MongoDB Atlas
* **APIs:**
  + Hugging Face Sentiment Analysis API
  + Whisper / Google Speech-to-Text (optional)
* **Scrapers:** GSMArena, Flipkart, Amazon

**5.3.4 Communications Interfaces**

* **HTTPS** for all communications
* **WebSockets / Socket.IO** for real-time dashboard updates
* **RESTful APIs** for module interactions

**6. Appendices**

This section contains supporting information that is **not essential to understanding the requirements**, but **provides reference material** useful for developers, researchers, and stakeholders.

**6.1 Glossary**

|  |  |
| --- | --- |
| Term | Definition |
| AI Persona Matching | A system that clusters users based on preferences and behaviors to recommend the best devices. |
| XAI (Explainable AI) | AI techniques like SHAP and LIME that provide human-understandable explanations for recommendations. |
| Cloud-Native | Applications designed to leverage cloud computing for scalability, resilience, and ease of deployment. |
| Microservices | Architectural style where the system is divided into small, independent services that communicate via APIs. |
| SLA (Service Level Agreement) | A performance guarantee in cloud services defining response times and availability. |
| Kubernetes (K8s) | Container orchestration platform for managing microservices and scaling cloud applications. |
| Prometheus/Grafana | Open-source tools for real-time monitoring and visualization of system metrics. |
| Gamification | Adding interactive elements like points, badges, and leaderboards to improve user engagement. |

**6.2 References**

[1] S. Sharma and S. Ghosh, "*Forecasting mobile prices: Harnessing the power of machine learning algorithms*," 2023 International Journal of Data Science and Analytics, vol. XX, no. XX, pp. XX–XX, 2023 - [Download](https://www.researchgate.net/publication/381392392_Forecasting_mobile_prices_Harnessing_the_power_of_machine_learning_algorithms)

[2] R. K. Das, A. P. Deshmukh and P. S. Deshmukh, "*Security and Privacy Problems in Voice Assistant Applications: A Survey*," IEEE Access, vol. 10, pp. 13820–13840, 2022 - [Download](https://arxiv.org/pdf/2211.01036)

[3] M. Tjoa and R. Guan, "*Explainable Artificial Intelligence (XAI) for Internet of Things: A Survey*," Computer Communications, vol. XX, no. XX, pp. XX–XX, 2021 - [Download](https://arxiv.org/pdf/2401.11441)

[4] A. Ray, B. Saha and M. Pal, "*Recommender Systems for the Internet of Things: A Survey*," IEEE Internet of Things Journal, vol. 8, no. 13, pp. 10325–10341, Jul. 2021 - [Download](https://dl.acm.org/doi/pdf/10.1145/3556536)

[5] A. Vartak, A. Karatzoglou and P. Covington, "*On-Device Recommender Systems: A Comprehensive Survey*," ACM Computing Surveys (CSUR), vol. 55, no. 5, pp. 1–38, 2023 - [Download](https://arxiv.org/pdf/2206.04800)

[6] M. Ahmad, M. A. Khan, S. A. Khan and M. H. Khan, "*Explainable AI over the Internet of Things (IoT): Overview, State-of-the-Art and Future Directions*," IEEE Access, vol. 10, pp. 20725–20750, 2022 - [Download](https://arxiv.org/pdf/2304.09486)

[7] D. Gunning and D. Aha, "*Understanding explainable artificial intelligence techniques: A comparative analysis for practical application*," Artificial Intelligence Review, vol. 55, no. 3, pp. 2303–2328, 2022 - [Download](https://archive.org/details/10.11591eei.v13i6.8378/page/4452/mode/2up)

[8] G. Shani and A. Gunawardana, "*Evaluating Recommender Systems: Survey and Framework*," in Recommender Systems Handbook, Springer, pp. 265–308, 2011 - [Download](https://dl.acm.org/doi/pdf/10.1145/3556536)

[9] K. Hightower, B. Burns and J. Beda, "*Kubernetes For Multi Cloud And Hybrid Cloud Orchestration*," ACM Queue, vol. 14, no. 4, pp. 70–93, Jul. 2016 - [Download](https://archive.org/details/kubernetes-for-multi-cloud-and-hybrid-cloud-orchestration/mode/2up?q=kubernetes)

[10] A. N. Sinha and R. K. Chauhan, "*Forecasting the Prices using Machine Learning Techniques: Special Reference to Used Mobile Phones*," Procedia Computer Science, vol. 198, pp. 516–523, 2022 - [Download](https://www.researchgate.net/publication/374107994_Forecasting_the_Prices_using_Machine_Learning_Techniques_Special_Reference_to_used_Mobile_Phones?utm_source=chatgpt.com)