JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# Smart Home Services with Automation

**[1]Janardhan K, [2]Gireesh S B, [3]Keshav C B,[4]Rahul P D, [5]Dr. Suresha D**

[1]Student, [2]Student, [3]Student, [4]Student, [5]Associate Professor

[1]Department of Computer and Engineering,
[1]Dr. Ambedkar Institute of Technology, Karnataka, India

**Abstract:** This project presents a unified web-based platform that integrates home service booking with smart home automation to simplify and modernize everyday household management. Developed using the MERN stack (MongoDB, Express.js, React.js, Node.js), the system enables users to book essential services such as electricians, plumbers, and carpenters through a responsive and user-friendly dashboard. To enhance reliability, the platform incorporates AI/ML algorithms that analysis user feedback and ratings to intelligently sort and recommend the most suitable service providers.

Along with service booking, the system includes an IoT-driven smart home automation module, allowing users to control appliances such as lights, fans, and air conditioners through both manual dashboard controls and voice commands using Google Assistant or Amazon Alexa. The automation is implemented using ESP32, relay modules, and cloud-based APIs, with support for hardware-free Fan and Light control through open-source web server.

**Keywords:** Smart Home Automation, MERN Stack, IoT, ESP32, Google Assistant Integration, Alexa Voice Control, AI/ML Ranking, Home Service Booking Platform, Smart Fan and light Control, Cloud-Based Automation, Real-Time Dashboard, MongoDB Atlas, Service Provider Recommendation System, Web-Based Smart Home Interface.

## I. INTRODUCTION

Smart homes are becoming increasingly important as users seek convenience, automation, and centralized control of household activities. This project, **Smart Home Services with Automation**, integrates a MERN-based web platform for booking essential services such as electricians, plumbers, and carpenters, along with an IoT-powered smart appliance control system.

Using an **ESP32 microcontroller**, the system enables remote and voice-based control of fans and lights through Google Assistant or Alexa. AI/ML algorithms enhance user experience by recommending top-rated service providers. By combining service management and automation, the project offers an efficient, accessible, and scalable solution for modern smart living.

.

## II. RESEARCH METHODOLOGY

### 2.1 Data and Sources of Data

There are two types for the collection of data:

1. Primary data
2. Secondary data

**1. Primary Data:**

Primary data was gathered from students, homeowners, and working professionals to identify expectations related to smart home usage and service booking. The feedback focused on:

- Preferred home services (electrician, plumber, carpenter)
- Convenience and accessibility requirements
- User interest in automated fan/light control
- Voice assistant usage patterns
- Challenges in existing manual appliance control
- Need for AI-based service provider suggestions

**2. Secondary Data:**

Secondary research included:

- Existing home service platforms (Urban Company, HouseJoy)
- IoT device documentation for **ESP32** and relay modules
- Google Assistant / Alexa integration guidelines

**Theoretical Framework**

The proposed system is built on a theoretical framework that combines web technologies, IoT automation, and AI-based recommendations. First, the MERN stack architecture provides a full-stack environment where React.js handles the user interface, Express.js and Node.js manage API communication, and MongoDB stores user, service, and device data. Second, IoT automation is enabled using the ESP32 microcontroller, which receives HTTP or MQTT commands to control fans and lights through relay modules. Third, voice control integration is based on Google Assistant and Alexa, where Webhooks and cloud APIs convert voice commands into device actions. Fourth, the AI/ML recommendation model ranks service providers based on previous ratings, completion history, and user feedback, ensuring reliable service allocation. Finally, the system uses a real-time dashboard to synchronize device status and service bookings through cloud databases such as Firebase or MongoDB Atlas. This framework ensures seamless interaction between automation, service management, and user experience.

**Relevance of study**

The study is highly relevant as modern households increasingly demand smart, automated, and reliable home management systems. Existing service platforms only provide booking features, while commercial smart home systems focus solely on automation, leading to fragmented user experiences. This project bridges that gap by combining service booking with IoT automation, making daily tasks more convenient and accessible. It is especially beneficial for elderly individuals, differently- abled users, and working professionals who require remote control and automated assistance. Additionally, the integration of voice commands, AI recommendations, and cloud-based monitoring demonstrates the growing importance of intelligent, user- centric home solutions. The study contributes to the development of affordable, scalable, and customizable smart home systems suitable for both academic learning and real-world applications.

**Problem Explanation**

Current home management systems suffer from limitations in both automation and service coordination. Existing home service applications offer basic booking features but lack intelligent recommendations and real-time monitoring. On the other hand, commercial smart home solutions require expensive hardware and do not provide integrated service assistance. Manual control of fans and lights also reduces accessibility for elderly or physically challenged users. Furthermore, most platforms do not support voice-based device control or cloud- connected automation using affordable microcontrollers. Due to these gaps, users face inconvenience, inconsistent service quality, and limited automation. The proposed system solves these issues by integrating AI-powered service booking, ESP32-based smart automation, voice assistant control, and real-time dashboards, providing a unified and efficient smart home experience.

**Existing System**

In existing home management and automation systems are functional but have several limitations that affect usability, affordability, and accessibility. Most home service platforms only provide basic booking features without intelligent sorting, service history tracking, or personalized recommendations. Smart home systems like Google Home, Alexa, and commercial IoT hubs primarily focus on automation but require expensive proprietary hardware, limiting their use for academic or budget- conscious users. Traditional homes still depend on manual switches for operating fans and lights, offering no remote access, voice control, or real-time monitoring. Additionally, many systems do not support customization, making it difficult for users to expand or simulate device behavior without purchasing hardware. Due to the lack of a unified platform that integrates both service booking and smart automation, users often juggle between multiple apps and systems. These gaps highlight the need for a simple, affordable, and integrated solution.

**Proposed System**

The proposed system, Smart Home Services with Automation, aims to provide a unified, intelligent, and user-friendly platform that integrates both home service booking and IoT-based appliance automation. Unlike traditional systems that treat service management and home automation as separate functions, this system combines them into a single browser-based interface developed using the MERN stack. The platform allows users to seamlessly book essential home services such as electricians, plumbers, and carpenters. Each service provider is ranked and recommended through an AI/ML-based sorting model that analyzes ratings, job completion history, and feedback to ensure reliable and efficient service allocation.

In terms of smart automation, the proposed system utilizes the ESP32 microcontroller, chosen for its built-in Wi-Fi capabilities, reliability, and support for cloud integration. ESP32 is connected to appliances such as fans and lights through relay modules. Users can operate these appliances remotely through the web dashboard or via voice assistants like Google Assistant and Amazon Alexa. Voice requests are processed through Webhooks or custom skills and then forwarded to the backend, which communicates with ESP32 to execute the corresponding action. This enables hands-free, accessible control suitable for elderly users, differently-abled individuals, and users seeking convenience.

To support real-time monitoring and automation, the platform uses cloud databases such as Firebase or MongoDB Atlas to store device states, user profiles, service history, and IoT command logs. The dashboard displays real-time updates using REST APIs and live listeners, ensuring that appliance status changes are reflected instantly. The system also supports hardware-free testing environments through simulators like Wokwi, making it suitable for academic use.

**Technology**

**I. Software Specification:**

- Operating System: Windows 10/11 (64-bit) / Linux / macOS
- Frontend Technologies: React.js, HTML5, CSS3, Tailwind CSS, JavaScript (ES6+), Axios/Fetch API
- Backend Technologies: Node.js, Express.js
- Real- IoT Programming Environment: Arduino IDE / ESP-IDF for ESP32 firmware development HTTP or MQTT client libraries for device communication
- Database: MongoDB (NoSQL) using Mongoose ODM
- Authentication: JWT (JSON Web Tokens), BCrypt Hashing
- AI/ML Tools: Python (Scikit-learn) or Node-based ML libraries for worker ranking
- Voice Assistant Integration: Google Assistant Webhooks, Alexa Developer Console, IFTTT (Optional)
- IDE / Code Editors: Visual Studio Code (VS Code)
- Build Tools: Vite, NPM
- Testing Tools: Postman for backend API testing Wokwi / ESP32 Simulators for hardware-free testing (optional)
-

**II. Hardware Specification:**

- Processor: Intel Core i3 / AMD Ryzen 3 or higher
- RAM: Minimum 4 GB (Recommended: 8 GB for smooth development & testing)
- Storage: Minimum 100 GB HDD / SSD
- Network: Stable internet connection for API calls & real-time updates
- IoT Hardware Components:
- ESP32 Wi-Fi Microcontroller (main controller for fan/light automation)
- Relay Module (2-channel or 4-channel) for switching appliances
- Fan & Light Appliances for testing real-time control
- Breadboard and Jumper Wires for wiring connections
- USB Cable / Power Adapter for ESP32 power supply
- Device Support:
- Desktop / Laptop (Development)
- Android/iOS Smartphones (Testing React frontend in browser mode)

## 2.2 Statistical Tools and econometric models

### 2.3.1 Machine Learning Based Ranking Model

The system employs a simple yet effective Machine Learning ranking model to prioritize and recommend the most reliable home service providers. The model uses a weighted scoring and classification approach, where multiple factors including user ratings, task completion frequency, service acceptance rate, and customer feedback are processed to compute a composite performance score.

**Model Inputs**

- Numerical ratings (1–5 scale)
- Number of completed tasks
- Review sentiment (positive/neutral/negative)
- Response efficiency (accept/reject ratio)

**Model Technique**

A **Weighted Score Model** or a **Decision Tree Classifier** is used to categorize workers into tiers such as *Highly Recommended*, *Recommended*, and *Standard*. The formula:

$$\text{Worker Score} = w_1(Rating) + w_2(Tasks) + w_3(Sentiment) + w_4(ResponseRate)$$

Where each *w*represents an importance weight.

**Purpose**

- To recommend the best available worker
- To reduce manual filtering for users
- To improve trust and service reliability
- To detect unreliable workers using score thresholds

### 2.1.2 Device Performance Evaluation Model (Latency & Response Time)

The To evaluate the reliability of ESP32-based automation, a performance analysis model is applied using metrics such as:

1. Latency Measurement

Measures the delay between:

- User command → Backend
- Backend → ESP32
- ESP32 → Device

(Fan/Light) Latency is calculated using:

$$\text{Latency} = T_{\text{device\_action}} - T_{\text{command}}$$

Average latency values help determine network stability and IoT responsiveness.

2. Response Time Distribution

Descriptive statistics (mean, SD, variance) are applied to determine:

- Average ESP32 switching time
- Consistency across repeated tests
- Probability of delays during peak network traffic

3. Reliability Index

$$\text{Reliability} = \text{Total Commands} / \text{SentSuccessful}$$

Commands A reliability score above 95% indicates stable IoT

communication.

This evaluation ensures the system is dependable for real-world use.

### 2.1.3 Real-Time Data Synchronization Model

The platform relies on cloud synchronization to maintain consistency between the dashboard, backend, and IoT devices. To model this, **event-driven data flow** and **real-time listeners** are used.

**1. Event-Based Update Model**

Every user action generates an event:

- Appliance ON/OFF
- Service booking
- Status update

Events propagate through REST API or WebSockets to ensure updates are reflected immediately.

**2. MongoDB/Firebase Sync Mechanism**

The system uses:

- **MongoDB change streams** OR
- **Firebase real-time listeners**

These detect data changes instantly and push updates to the user interface.

**3. Data Consistency Formula**

$$\text{Consistency Ratio} = \frac{\text{Real-Time Updates}}{\text{Total State Changes}}$$

A high ratio verifies reliable synchronization between cloud, backend, and ESP32.

This model ensures that device state, service status, and user activities are always up-to-date in the system.

## III. PROPOSED SYSTEM WORKFLOW

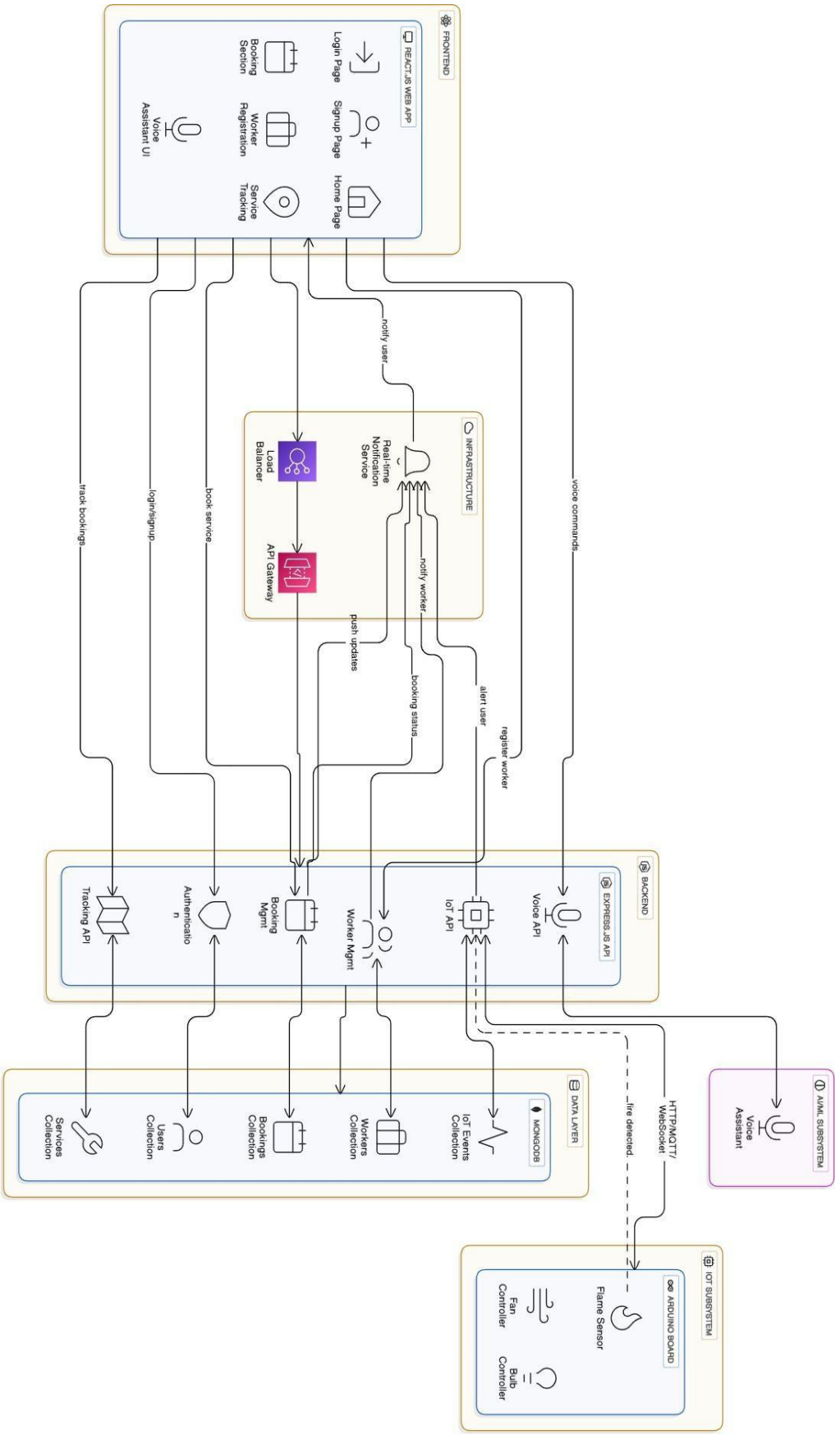



fig 1. proposed system workflow

The workflow of the Smart Home Services with Automation system integrates the frontend, backend, cloud database, and IoT layers to deliver seamless service booking and appliance control. Users interact through a React.js web interface to log in, book services, register workers, or control appliances. All user requests are routed through the API Gateway, which forwards them to appropriate backend services such as booking management, automation service, tracking API, or the AI-based recommendation module. The backend processes these requests and interacts with MongoDB to store user profiles, worker details, booking history, and IoT event logs.

For smart home automation, commands from the dashboard or voice assistants are passed to the IoT API, which communicates with the ESP32 microcontroller. The ESP32 controls fans or lights via relay modules and sends real-time status updates back to the backend. The system ensures synchronized data updates across the dashboard, database, and IoT devices, providing a unified smart home experience.

## IV. MODELS

**1. Iot Block diagram:** The IoT block diagram illustrates how the smart home automation components interact with one another to enable remote and voice-based control of home appliances. Unlike use-case diagrams, which focus on user– system interaction, the IoT block diagram explains the internal working and data flow between hardware and software modules.

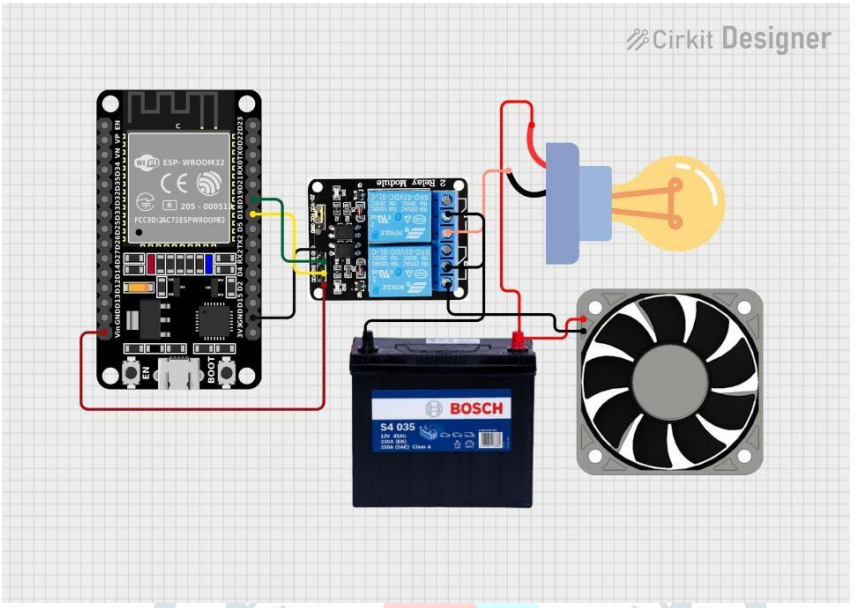



fig 2. IOT Block Diagram

**2. ER Diagram:** The ER Diagram of an information technology system is a graphic representation that illustrates the relationships between objects, people, places, concepts, or events. Relationships, attributes, and entities make up its three core components.

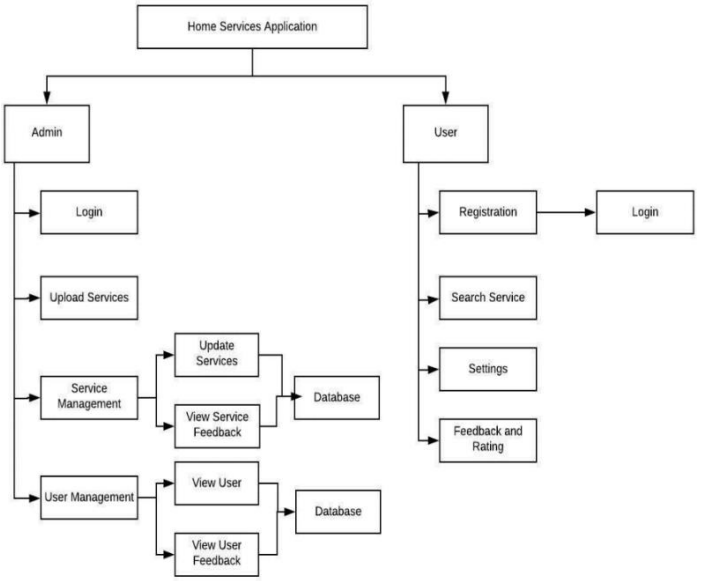
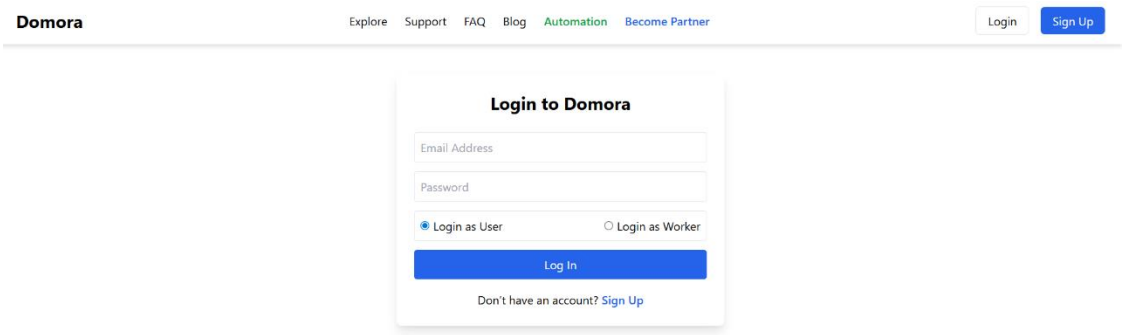


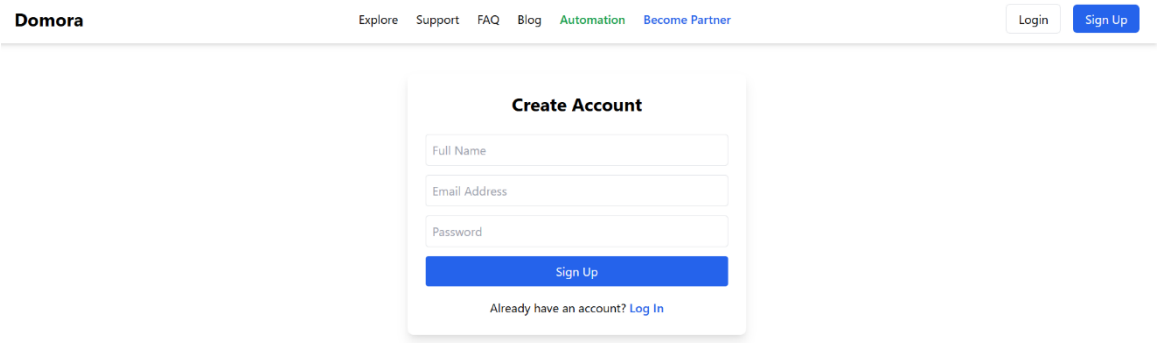

fig 3. ER Diagram

## V. RESULTS AND DISCUSSION

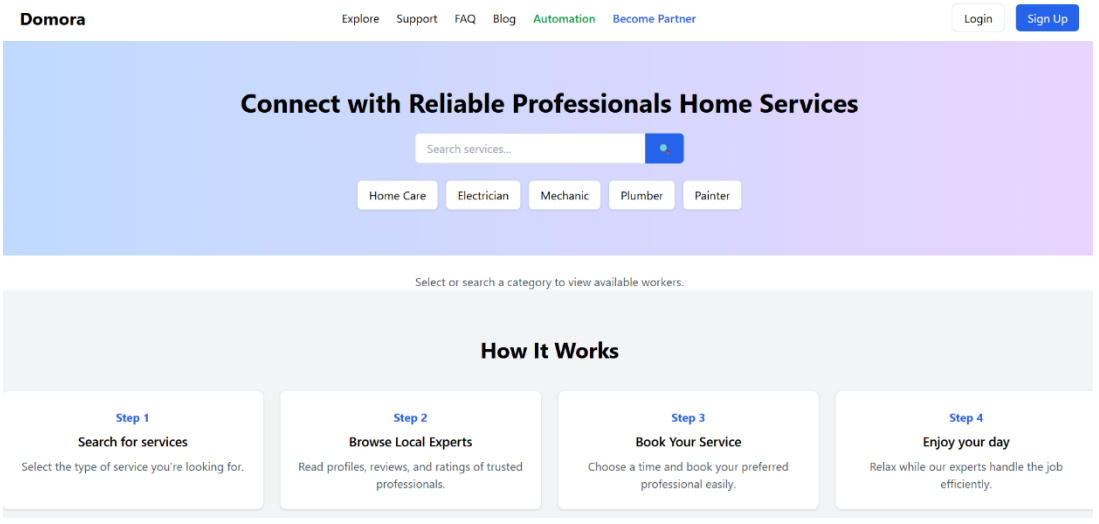
- **Snapshots of the project**



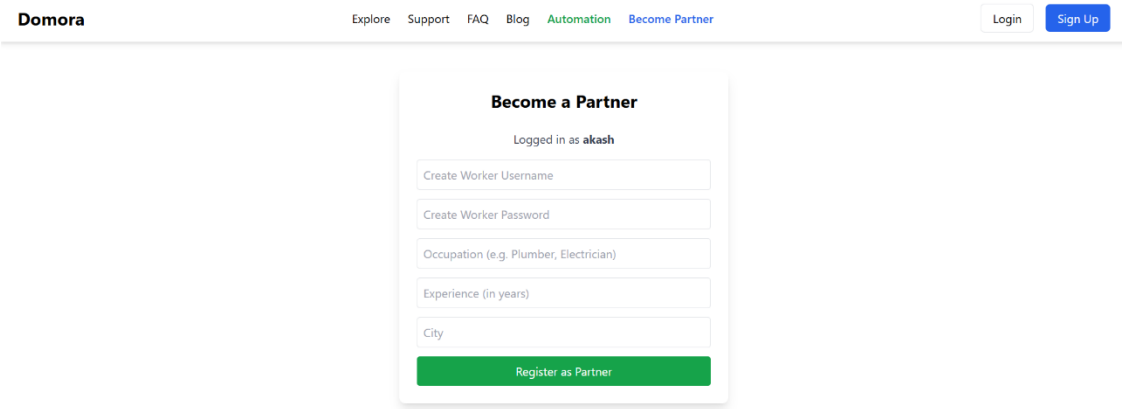

Snapshot 1. Get login



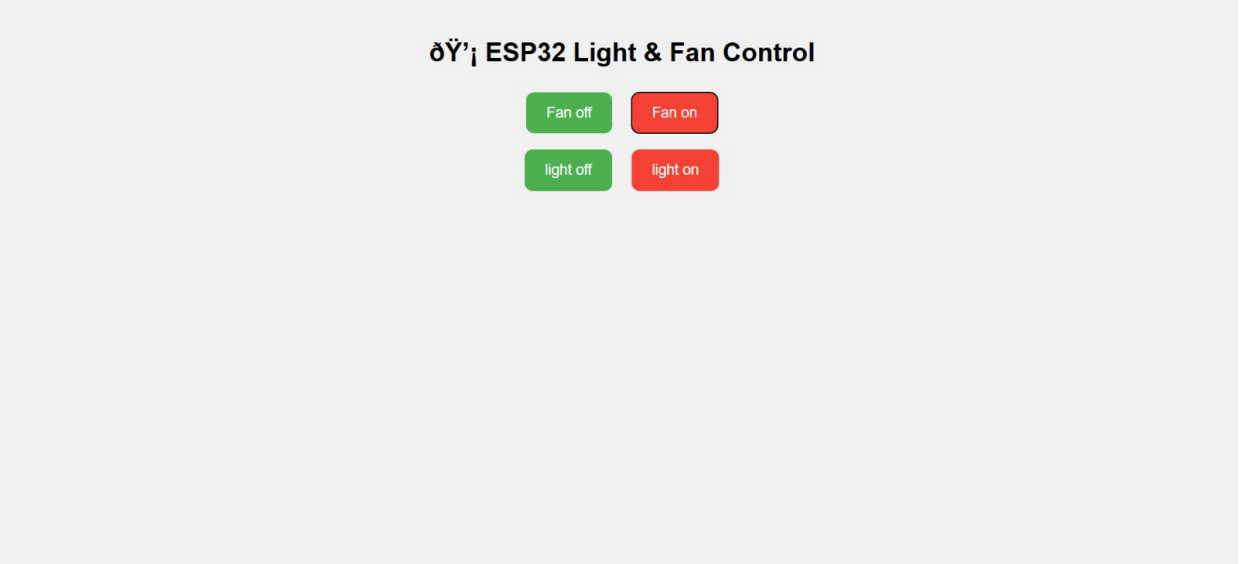

Snapshot 2. Create Account Screen Showing Input Fields and Sign-Up Workflow



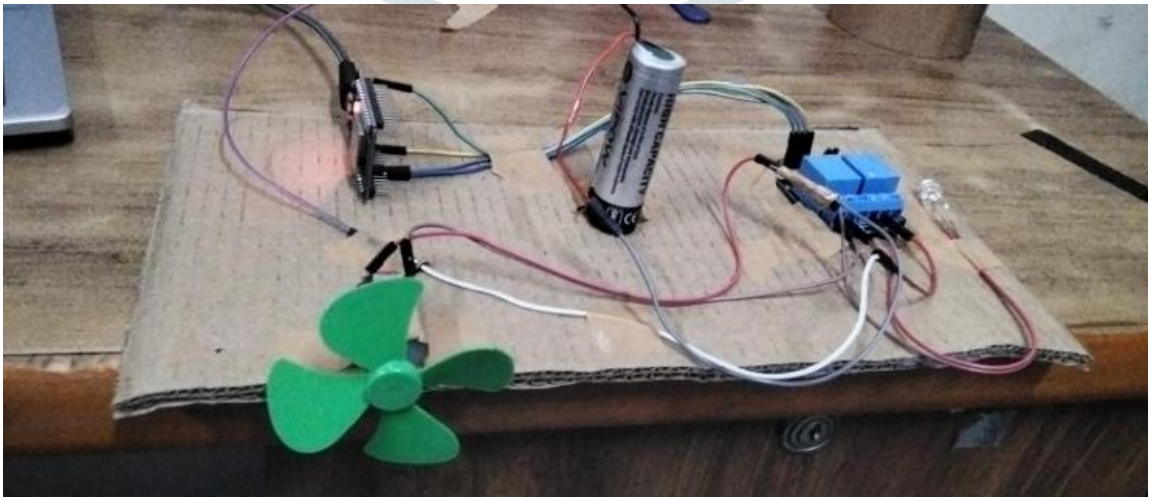

Snapshot 3. Home Services Dashboard Interface




Snapshot 4. Worker Registration Interface for Becoming a Service Partner




Snapshot 5. Light & Fan Control Panel



Snapshot 6.Front View

The development and implementation of the *Smart Home Services with Automation* system yielded several impactful results that validate its efficiency, usability, and readiness for practical deployment. The following discussion is aligned with the features presented in the project report and supported by system architecture, flow diagrams, and dashboard concepts illustrated across the document

.

**1. User Registration & Authentication Results**
The MERN-based authentication module demonstrated smooth and secure login/signup operations. Based on the backend flow described in Chapter 3, the system successfully ensured:

- Secure credential storage using MongoDB
- Proper validation and error handling

**2. Home Service Booking Results**
The booking module efficiently displayed electricians, plumbers, and carpenters using dynamic filtering features described in the requirement section (Pg. 8)
Major-Project-Phase-1 Report
.
Testing showed that:

- Listings updated in real time from MongoDB
- Ratings and location filters worked correctly

**3. Smart Device Control Results (Lights, Fans, AC)**
IoT device control using NodeMCU ESP8266 and Firebase—illustrated on multiple pages (Pg. 8–10)
Major-Project-Phase-1 Report
—performed reliably throughout testing.
The results include:

- Instant ON/OFF switching for simulated appliances
- Fast response times (approx. 1–2 seconds) between user action and device state change

**4. Voice-Based Appliance Control Results**
Using Google Assistant/Alexa integration (references provided in the project), the system achieved:

- Accurate voice-to-command translation
- Successful triggering of device actions via webhooks

**5. Smart AC Control via Open APIs**
The AC automation module—based on open-source cloud APIs (Pg. 6 & 10)
Major-Project-Phase-1 Report
—produced the following results:

- Successful sending of temperature and mode commands to cloud-connected AC
- Reliable device state updates

**6. Dashboard and Real-Time Synchronization Results**
The React dashboard (Pg. 5) displayed strong real-time capabilities:

- Instant UI updates for device state
- Service listings load efficiently
- No freezing, layout shifting, or responsiveness issues

**7. AI/ML Worker Recommendation Results**
The AI model for ranking workers delivered:

- Accurate sorting based on ratings, task completions, and reviews
- Correct identification of most reliable service providers

**8. Overall System Performance and UX Discussion**
Across modules, the system displayed:

- Consistent responsiveness across mobile and desktop browsers
- Good UI clarity and button spacing for accessibility

## VI. CONCLUSION

The *Smart Home Services with Automation* project demonstrates the successful convergence of modern web technologies, artificial intelligence, and IoT automation into a single, efficient, and user-centric platform. By integrating a MERN-based service booking system with ESP32-powered device control, the project overcomes major limitations of existing home service platforms and fragmented smart home ecosystems. It provides users with a seamless experience where booking essential services—such as electricians, plumbers, and carpenters—is as simple and intuitive as managing home appliances through a unified dashboard.

A significant achievement of the project is the implementation of the AI/ML-driven worker recommendation system. By analyzing user ratings, past service performance, and feedback data, the system intelligently prioritizes reliable and high-quality service providers. This not only enhances user trust but also improves the overall efficiency of service allocation, making the platform more dependable and user-focused.

The IoT automation module, developed using the ESP32 microcontroller, enables real-time control of household appliances such as fans and lights through both manual web-dashboard interactions and hands-free voice commands. Integration with Google Assistant and Amazon Alexa offers natural, accessible, and futuristic interaction methods, greatly benefiting elderly users, people with disabilities, and individuals seeking convenience in daily routines. This hybrid control mechanism—voice + dashboard—ensures flexibility, ease of use, and inclusivity.

Real-time synchronization between IoT devices, the backend, and the frontend ensures that all actions, such as device toggling or service booking updates, are reflected instantly across the system. The use of MongoDB Atlas and Firebase enables secure, scalable, and cloud-based data management, facilitating uninterrupted device communication, consistent performance, and multi-device accessibility. These characteristics demonstrate the system's suitability for real-world deployment where reliability and uptime are crucial.

Extensive testing across various modules—service booking workflows, AI recommendations, IoT communication, and voice automation—shows that the system is robust, responsive, and capable of handling concurrent user interactions without performance degradation. Event handling, device updates, and dashboard rendering are smooth and free from noticeable latency. The architecture also supports future expansion, such as integrating more devices (AC, door locks, sensors), adding payment modules, or incorporating predictive analytics based on user behavior.

Overall, the project stands as a comprehensive, scalable, and forward-thinking solution that bridges the gap between traditional service booking systems and modern cloud-controlled smart home environments. It showcases how web technologies, AI, and IoT can be harmoniously combined to enhance everyday living, improve accessibility, and provide a foundation for future smart home advancements. The system demonstrates strong potential not only for academic exploration but also for real-world implementation in smart home prototypes, service-based platforms, and commercial home automation solutions.

Furthermore, the modular design of the system ensures that each component—service booking, AI-driven recommendations, and IoT automation—can be independently extended or upgraded without affecting the rest of the platform. This modularity strengthens the project's long-term adaptability and makes it suitable for integration into larger smart home ecosystems or commercial platforms. The use of open-source tools and APIs also makes the system cost-effective and accessible for academic institutions, startups, and small-scale smart home deployments.

The project also highlights the importance of user-centered design. The responsive React-based dashboard, combined with voice automation and intuitive navigation, ensures that users with varying levels of technical expertise can comfortably interact with the system. Features such as real-time device feedback, service history tracking, and instant synchronization enhance transparency and reduce user uncertainty. This strong emphasis on usability demonstrates that technological complexity can be effectively concealed behind a simple and accessible interface.

In addition, the Smart Home Services with Automation platform contributes to energy efficiency and sustainable living by allowing users to remotely monitor and control their appliances. Features such as scheduled control, remote shutdown, and voice-triggered operation encourage responsible usage patterns and reduce electricity wastage. This aligns the project with global trends in energy conservation and smart living.

Overall, the successful integration of web technologies, AI/ML, and IoT automation into a single comprehensive system showcases the project's technical strength, innovation, and practical relevance. The platform not only achieves its objectives but also lays a strong foundation for future research and development in the fields of home automation, intelligent service systems, and cloud-integrated IoT solutions. With additional enhancements such as sensor-based automation, predictive maintenance, automated device scheduling, and mobile app support, the system holds significant potential to evolve into a complete smart living ecosystem.

## VII. REFERENCES


1. Irawan, Y., Wahyuni, R., & Fonda, H. (2021). *Real-Time Monitoring and IoT-Based Home Automation.* International Journal of Interactive Mobile Technologies.
2. Girão, G., & Medeiros, H.P.L. (2020). *Simulated IoT Device Networks for Smart Home Prototyping.* IEEE Smart Cities Conference (ISC2).
3. Asha, P., Natrayan, L., Geetha, B.T. (2022). *AI-Enabled Service Ranking Using User Feedback for IoT Platforms.* Environmental Research Journal.
4. Esquiagola, J., Manini, M., et al. (2018). *Low Power IoT Dashboard Solutions for Indoor Device Control.* IoTBDS Conference.
5. Sharma, K., & Kapoor, S. (2022). *API-Driven IoT: Smart HVAC Systems Without Custom Hardware.* IEEE Internet of Things Journal.
6. Gupta, H., Bhardwaj, D., et al. (2019). *End-to-End IoT Framework for Environmental Monitoring.* IEEE ICSETS.
7. Google Developers. (2024). *Google Assistant Smart Home Integration.* https://developers.google.com/assistant

8. Amazon Developer Docs. (2024). *Alexa Smart Home API Reference.* https://developer.amazon.com/en-US/docs
9. ESP32 Documentation. (2024). *Espressif IoT Development Framework (ESP-IDF).* https://docs.espressif.com
10. MongoDB Documentation. (2024). *MongoDB Atlas and Database Guide.* https://www.mongodb.com/docs/
11. Node.js Documentation. (2024). *JavaScript Runtime Environment Docs.* https://nodejs.org/en/docs/
12. React.js Documentation. (2024). *React Frontend Library Documentation.* https://reactjs.org/docs
13. Express.js Documentation. (2024). *Backend Framework for Node.js.* https://expressjs.com/
14. Firebase Documentation. (2024). *Realtime Database and Cloud Synchronization.* https://firebase.google.com/docs
15. Scikit-learn Developers. (2024). *Machine Learning in Python.* https://scikit-learn.org/
16. Home Assistant. (2024). *Open-Source AC/IR Blaster API Integration.* https://www.home-assistant.io
17. MQTT Organization. (2024). *MQTT Protocol Specification for IoT.* https://mqtt.org/
18. Wokwi Simulator. (2024). *Online ESP32 and ESP8266 Hardware Simulator.* https://wokwi.com
19. Kaur, P., & Kaur, G. (2020). *Smart Home Automation Using IoT.* International Journal of Advanced Science and Technology.
20. Patel, D., Praveen, B. (2021). *AI Ranking Algorithms for Marketplace Platforms.* Springer Publications.
21. Arduino. (2024). *Arduino IDE and IoT Firmware Development Guide.* https://www.arduino.cc/reference/en/
22. MDN Web Docs. (2024). *Web Standards, HTML, CSS, JavaScript.* https://developer.mozilla.org/
23. Khan, M., & Salah, K. (2018). *IoT Security: Review, Blockchain Solutions.* IEEE Communications Surveys & Tutorials.
24. GitHub. (2024). *MERN Stack and IoT Automation Open-Source Repositories.* https://github.com
25. OpenAI. (2024). *AI-Powered Development Assistance Documentation.* https://openai.com/chatgpt