Developed By-

Sanket Wagh | Anish Patil

Software Requirements Specification (SRS)

Project: SmartOccupy – Smart Crowd Monitoring & Suggestion System

Tagline: Know the Crowd Before You Go

Developed By: Anish Patil | Sanket Wagh

---

1. Introduction

1.1 Purpose

The purpose of this SRS is to define the functional and non-functional requirements of *SmartOccupy*, a full-stack IoT-based crowd monitoring and suggestion system for malls and retail stores.
The document outlines system behaviour, constraints, interfaces, and performance expectations.

1.2 Scope

SmartOccupy monitors real-time footfall using IR sensors and ESP32 modules installed at store entry and exit points.
The system provides:

- **Admins (Store Owners):** Live occupancy dashboard, analytics, threshold alerts
- **Customers:** Live crowd map, store navigation, safer shopping decisions

It is built using Next.js, Clerk, MongoDB, and ESP32 sensor devices.

1.3 Definitions

- **Footfall:** Number of people entering/exiting a store
- **Occupancy:** Current number of people inside a store
- **IoT Device:** ESP32 with dual IR sensors
- **RBAC:** Role-Based Access Control (Admin, Customer)

---

2. Overall Description

2.1 Product Perspective

SmartOccupy is an integrated IoT + Web application system consisting of:

- Hardware layer (IR sensors + ESP32)
- Network layer (WiFi + HTTP)
- Application layer (Next.js Web App)
- Database layer (MongoDB)

The system updates occupancy every time a sensor event occurs.

2.2 Product Features

- Real-time crowd counting
- Visual color-coded map (Green/Yellow/Red)
- Admin analytics dashboard
- Store capacity management
- Sensor-device mapping
- Crowd alerts

2.3 User Classes & Characteristics

Admin (Shop Owner)

- Registers store and IoT device ID
- Monitors live crowd
- Sets store capacity
- Views analytics

Customer

- Searches stores
- Views crowd density
- Navigates to stores

2.4 Operating Environment

- **Frontend:** Browser (Chrome, Edge, Safari)
- **Backend:** Next.js 15 API on Node.js
- **Database:** MongoDB Atlas
- **Hardware:** ESP32, IR Sensors, WiFi

---

3. System Features

3.1 Feature: Real-Time Occupancy Counter

Description

Counts entries/exits using dual IR sensors and updates store occupancy.

Functional Requirements:

- FR1: System must receive entry or exit signals via /api/iot/update.
- FR2: System shall update the store's current occupancy.
- FR3: System shall prevent negative occupancy values.

3.2 Feature: Admin Dashboard

Description

Displays real-time dashboards with alerts.

Functional Requirements:

- FR1: Admin shall view current occupancy in real-time.
- FR2: System shall change dashboard indicator to RED if occupancy > 80%.
- FR3: Admin shall update store name, capacity, and device ID.

3.3 Feature: Customer Map View

Description

Customers view store crowd levels on a map.

Functional Requirements:

- FR1: System shall show stores on map with color-coded markers.
- FR2: Customers can filter stores by name, category, or floor.
- FR3: Clicking "Navigate" opens Google Maps.

3.4 Feature: Footfall Analytics

Functional Requirements:

- FR1: System shall maintain hourly footfall logs.
- FR2: Analytics data shall be provided via /api/analytics.

---

4. External Interface Requirements

4.1 User Interface

- Clean UI using Shadcn/UI + Tailwind
- Mobile and desktop responsive
- Map interface powered by Leaflet

4.2 Hardware Interface

- ESP32 sends JSON:

```
{
  "deviceId": "store123",
```

"entry": 1,

"exit": 0

}

4.3 Software Interfaces

- MongoDB
- Clerk Authentication
- REST API Endpoints

---

5. System Architecture

Data Flow:
IR Sensor → ESP32 → Next.js API → MongoDB → Real-Time Dashboard

Components:

- Sensor Module
- API Layer
- Database Layer
- Web Dashboard
- Customer Map System

---

6. API Requirements

Method	Endpoint	Description
POST	/api/iot/update	Receive sensor data from ESP32
GET	/api/stores	Public store list
GET	/api/stores/mine	Admin-specific store data
GET	/api/analytics	Occupancy + footfall analytics

---

7. Non-Functional Requirements (NFRs)

7.1 Performance

- API response time < 300ms
- Dashboard refresh < 1s

7.2 Security

- Clerk authentication (JWT)

- RBAC enforced
- Sensor requests validated by deviceId

7.3 Reliability

- System shall handle missed sensor pulses
- Auto-retry for ESP32 HTTP failures

7.4 Scalability

- Support up to 500 stores per mall
- Handle 1000 requests/minute

7.5 Availability

- 99% uptime (cloud deployment)

---

8. Constraints

- Internet dependency for ESP32
- IR sensors may misread during extreme lighting
- Deployment requires stable WiFi

---

9. Future Enhancements

- AI Camera-Based Footfall Detection
- Push Notifications for crowd alerts
- Heatmap visualization
- Multi-mall integration

---

10. Appendix

- ESP32 wiring diagrams
- JSON response templates
- Store schema sample

Developed By:

Anish Patil | Sanket Wagh