

# Pattern Detector Integration – Product Requirements Document (PRD)

## 1. Introduction

### 1.1 Purpose

Extend the existing AI-First Internal Helpdesk Portal to include a Pattern Detector component. This feature must autom

### 1.2 Scope

The Pattern Detector will:

- Continuously analyze newly created tickets (title + description + metadata).
- Group “similar” tickets into clusters when they share high textual or metadata similarity.
- Flag clusters that exceed configurable thresholds (e.g.,  $\geq N$  tickets about “VPN down” in T minutes).
- Provide endpoints and dashboard widgets for:
  - Listing current clusters and their member ticket IDs.
  - Raising an “Incident” parent ticket (optional) covering a cluster.
  - Issuing notifications or alerts to department leads.
  - Deprioritizing or holding suspected spam/misuse tickets for manual review.

### 1.3 Definitions

- Ticket: A record submitted by a General User (fields: ticketId, title, description, department, priority, createdAt, etc.).
- Cluster: A set of tickets grouped by similarity (fields: clusterId, keywords, memberTicketIds, firstSeen, lastSeen, depar
- Incident Ticket: A special ticket created (optionally) to represent a recurring issue affecting multiple users (fields: inci

## 2. Objectives and Goals

### 1. Reduce Redundancy

- Automatically group multiple tickets reporting the same underlying issue (e.g., “VPN not connecting,” “VPN error,”

### 2. Proactively Alert

- When a cluster’s size crosses a department-specific threshold (e.g., 5 “VPN” tickets in 10 minutes), send an alert to

### 3. Enable Incident Management

- Optionally create a parent “Incident” ticket that aggregates all member tickets and allows agents to post a single res

### 4. Prevent Misuse/Spam

- If a single user or unusual pattern of repeat submissions is detected (e.g., one user opening  $> 3$  “test” tickets in 5 mi

### 5. Provide Analytics

- Expose real-time “Top Recurring Issues” metrics and “Pattern Alerts” in department dashboards.

## 3. Functional Requirements

### 3.1 Data Ingestion & Preprocessing

#### - Trigger Point

- Every time a new ticket is created (POST /tickets), its title, description, and metadata (userId, createdAt, department)

#### - Preprocessing Steps

1. Normalize Text: Lowercase, strip punctuation, remove stop words (common words like “the,” “and”).
2. Extract Keywords: Identify nouns/verbs and domain-specific terms (e.g., “VPN,” “printer,” “leave policy”).
3. Compute Embedding: Use the same embedding model (e.g., OpenAI embeddings) as Response Suggestion to obtain

### 3.2 Clustering and Similarity

#### - Similarity Metric

- Compute cosine similarity between the new ticket’s embedding and existing active clusters’ centroid embeddings.
- A ticket “matches” an existing cluster if similarity  $\geq 0.85$  AND both have the same department value (unless departm

#### - Cluster Membership Logic

1. If a matching cluster exists, add ticketId to that cluster’s memberTicketIds. Update lastSeen = now. Recompute the
2. If no existing cluster meets similarity threshold, create a new cluster object with: