# Problem Statement 12

## Website Health Monitor with Multi-Channel Alerts (Django)

Reference:HTTP semantics RFC 9110 (and 9112/9113/9114), JSON RFC 8259, ICMP RFC 792/4443, TLS RFC 8446, SMTP RFC 5321/5322/6409/4954/3207/8314, WebSocket RFC 6455, Web Push RFC 8030/8292; and W3C: Web Notifications, Push API, Service Workers, WebSocket API.

**Objective**

Build a **Django web application** that monitors a target website's health by periodically pinging its URL. If the site remains **down for more than 2 minutes**, notify users via:

- **In-app notifications** (AJAX refresh; no full page reload)
- **Email** (SMTP)
- **SMS** (Twilio)

**Problem**

Service downtime hurts reliability and user trust. Teams must implement a system that:

1. **Continuously monitors** a configurable target URL.
2. Detects when the site is **unavailable for > 2 minutes** and **raises a notification event**.
3. Delivers notifications to all registered users across three channels:

    - **In-App:** AJAX-driven notification dropdown.
    - **Email:** Django SMTP backend.
    - **SMS:** Twilio API (test credentials acceptable).

**Requirements**

**1) Backend (Django)**

- **Models**

    - User: username, email, mobile
    - Notification: title, message, status (read/unread), channel (in-app/email/sms), timestamps

- **Monitoring**

    - A ping/check function running **every 30 seconds** (e.g., Celery beat/CRON/thread) recording up/down state and timestamps
    - Logic to detect **continuous downtime ≥ 120 seconds** before triggering notifications

- **APIs**
    - Endpoint(s) to **fetch unread notifications**
    - Endpoint to **mark as read**

**2) Frontend (HTML, Bootstrap, JS, AJAX)**

- **Navbar notification dropdown** showing unread count + list
- **AJAX polling** (e.g., every 5–10s) to fetch new notifications without page reload
- **Mark-as-read** interaction (single or bulk)

**3) Notification Channels**

- In-App: Store Notification rows; expose via JSON; render in dropdown
- Email: Use Django's SMTP (send_mail or EmailMessage)
- SMS: Use Twilio (twilio.rest.Client) with test credentials; handle failures gracefully

**Configuration & Security**

- Target URL, polling interval, and alert threshold should be **configurable**
- Use **environment variables** for SMTP/Twilio secrets
- Basic **rate-limiting**/deduping to avoid alert storms (e.g., send once per outage until recovery)

**Bonus (Optional)**

- **Recovery notifications** when site comes back up
- **Simple dashboard**: uptime %, last outage, MTTR
- **Webhooks** (Slack/Teams) as additional channels

**Evaluation Criteria**

- **Ping check correctness** (30s cadence; state tracking)
- **Downtime threshold** (> 2 minutes) reliably triggers notifications
- **In-app UX** updates via AJAX (no full reload)
- **Email delivery** via SMTP
- **SMS delivery** via Twilio test creds
- **Code quality**: structure, comments, README with setup/run steps