Team Name: DNinjaS

Members:

- Sarthak Santosh Dhamale(Student)
- Ayush Harichandra Nakhale (Student)
- Bhagyashri Tanaji Thorat(Professor)

Problem Statement: Website Health Monitor with Multi -Channel Alerts (Django)ver NATS

TABLE OF CONTENTS

Blog link

Introduction

- **Theme:** Implementation and Testing of Selected Internet-Drafts / RFCs using AIORI Testbed
- **Focus Areas:** Website Health Monitor with Multi-Channel Alerts (Django) — based on HTTP (RFC 9110/9112/9113/9114), JSON (RFC 8259), ICMP (RFC 792/4443), TLS (RFC 8446), SMTP (RFC 5321/5322/8314), and Web Push & WebSocket (RFC 8030/6455).
- **Organized by:** Advanced Internet Operations Research in India (AIORI)
- **Collaborating Institutions:** International Institute of Information Technology, Pune / AIORI Anycast & Measurement Nodes Network
- **Date:**11/2025
- **Prepared by:**

Name	Designation	Institution
Ayush Harichandra Nakhale	Student	International Institute of Information Technology, Pune
Sarthak Santosh Dhamale	Student	International Institute of Information Technology, Pune
Bhagyashri Tanaji Thorat	Professor	International Institute of Information Technology, Pune

Contact: sarthak.s.dhamale@gmail.com 8799981639 **;** ayushnakhale251@gmail.com 9673572354

- ## Executive Summary

The project “Website Health Monitor with Multi-Channel Alerts” was developed under the AIORI 2.0 Hackathon to implement and test key IETF RFCs related to website availability, transport security, and real-time communication. The implementation focuses on building a Django-based system capable of continuous website health monitoring, automated outage detection, and multi-channel alerting (In-App, Email, and SMS).

Through this project, RFCs such as HTTP Semantics (RFC 9110/9112/9113/9114), ICMP (RFC 792/4443), TLS 1.3 (RFC 8446), SMTP (RFC 5321/5322/8314), JSON (RFC 8259), and Web Push (RFC 8030/8292) were implemented in a controlled test environment.

The contribution provides a working open-source prototype that integrates seamlessly with AIORI’s testbed for Internet measurement and monitoring, enabling real-time website status tracking and alerting across multiple channels. This aids in understanding Internet service reliability from end-user and node perspectives, aligning with AIORI’s mission of improving Internet resiliency and measurement infrastructure in India.

- ## Overview

Implement website monitoring RFCs in a controlled Django environment, contributing directly to open-source HTTP and notification tools. This work generates implementation feedback for future IETF standards on web reliability and push protocols, while building local developer capacity in network observability and automation.

- **Objectives**
  - Implement selected RFCs / Internet-Drafts related to website monitoring and communication protocols in a controlled Django environment.
  - Contribute improvements and documentation to open-source repositories relevant to HTTP monitoring, email handling, and notification systems.
  - Generate implementation feedback for future IETF work on web reliability, alerting systems, and push notification protocols.
  - Build local developer capacity in Internet standards implementation, particularly in network observability and alert automation.
- **Scope and Focus Areas**

Focus Area	Relevant RFCs / Drafts	Open-Source Reference	AIORI Module Used
Website Health Monitoring	RFC 9110–9114 (HTTP/1.1–HTTP/3), RFC 8259 (JSON)	Django REST Framework, Requests	AIORI HTTP Measurement Testbed
Network Reachability (Ping/ICMP)	RFC 792 (ICMPv4), RFC 4443 (ICMPv6)	Python ICMP module, AIORI anchor scripts	AIORI Latency & Availability Module
Transport Security	RFC 8446 (TLS 1.3)	Requests (HTTPS checks), OpenSSL	AIORI Secure Connectivity Module
Email Notification System	RFC 5321 / 5322 / 8314 (SMTP)	Django SMTP Backend	AIORI Communication Module
WebSocket & Push Alerts	RFC 6455 (WebSocket), RFC 8030 / 8292 (Web Push)	Django Channels, Service Worker API	AIORI Web Notification Module

- **Sprint Methodology**

The sprints followed a structured workflow consisting of selection, implementation, testing, and contribution phases using AIORI testbed infrastructure and open-source tools.

- **Workflow:**
  - RFC / Draft Selection
    - Relevant RFCs were selected based on their direct applicability to web health monitoring:
    - RFC 9110–9114 (HTTP/1.1–HTTP/3): Used for website status checks via Django backend.
    - RFC 792 / 4443 (ICMP): Implemented partially for ping-based uptime verification.
    - RFC 8259 (JSON): Used in AJAX responses to display UP/DOWN status dynamically.
    - RFC 8446 (TLS 1.3): Referenced for HTTPS validation, planned for secure-check feature.
    - RFC 5321 / 5322 (SMTP) and RFC 8314: Referenced for email alerts (backend setup incomplete).
- **Sprint Preparation:**
  - Frameworks Used: Django, Bootstrap 5, AJAX, Celery, Chart.js
  - Database: SQLite (default Django DB)
- **Environment:**
  - Python virtual environment, folder path C:\Users\sarth\OneDrive\Documents\AIORI2-Project12\FRONT\src
- **Version Control:**
  - Git initialized in /FRONT folder and connected to GitHub.
- **Testing Setup:**
  - Localhost-based monitoring of real and test URLs using Django views and AJAX requests
- **Implementation Phase:**

Sprint	Goal	Progress
Sprint 1	Set up Django models, base templates, and navigation pages	Completed
Sprint 2	Implement AJAX website UP/DOWN check using HTTP requests	Working successfully
Sprint 3	Integrate Celery for periodic background website checks	Basic working
Sprint 4	Build dashboard layout and design uptime graph interface	UI completed
Sprint 5	Notification channels (in-app/email/SMS)	In App Notifications implemented

- **Interoperability Testing:**
  Testing focused on verifying website status checks and Celery scheduling:
  - Confirmed HTTP status responses using requests module (RFC 9110 compliance).
  - AJAX integration successfully displayed real-time UP/DOWN status without page reload.
  - Celery tested for background task execution (interval runs confirmed, not yet RFC-timed).
  - Notification and alert channel tests deferred to next phase.

- **Documentation & Contribution:**
  - Each sprint committed and documented in the GitHub repository.
  - README updated with setup commands, environment info, and RFC references.
  - Weekly mentor meetings logged development progress and pending issues.
- **Post-Sprint Reporting:**
  - The frontend, website check system, and Celery monitoring loop were demonstrated as a working proof of concept with In App notifications.
  - Email/SMS alerts are part of upcoming implementation stages.
  - Integration readiness for AIORI testbed latency comparison noted for future testing phase.

- **Activities and Implementation**

Date	Activity	Description	Output / Repository
10/10/2025		Created the Django project, set up base templates (index.html, about.html, how_it_works.html), and configured	https://github.com/Starc6254/DNinjaS-Website-Health-Monitoring--AIORI-2-
13/10/2025	Sprint 2: Website Availability Check (HTTP)	Implemented real-time UP/DOWN website check using Python requests (RFC 9110) and AJAX JSON response	https://github.com/Starc6254/DNinjaS-Website-Health-Monitoring--AIORI-2-
18/10/2025	Sprint 3: Celery Background Monitoring	Integrated Celery to perform automatic background checks for website uptime and response time at fixed intervals	https://github.com/Starc6254/DNinjaS-Website-Health-Monitoring--AIORI-2-
22/10/2025	Sprint 4: Dashboard UI Design	Developed dashboard interface with Bootstrap 5 and Chart.js for visual uptime display (using demo data).	https://github.com/Starc6254/DNinjaS-Website-Health-Monitoring--AIORI-2-

- **Results and Findings**

During testing, the Website Health Monitoring (WebCheck) system successfully performed real-time website checks using HTTP requests and AJAX updates. The application accurately showed whether a site was UP or DOWN without reloading the page.

The Celery background task worked for periodic checks, but the timing intervals sometimes varied, which will need tuning. The dashboard UI and graph section were completed but currently display demo data instead of live results.

Some challenges included connecting notifications to live events and ensuring Celery followed exact intervals. Despite these, the system proved that Django and Celery can effectively handle website monitoring in line with key RFCs such as HTTP (RFC 9110) and ICMP (RFC 792).

Overall, the project achieved a functional prototype that can be extended with alerts, analytics, and AIORI testbed integration in future stages.

- **Open-Source Contributions**

The project code was maintained in a local Git repository throughout the development phase, with regular commits for each sprint. The repository includes Django source files, Celery scripts, AJAX integration, and frontend templates.

While no external pull requests were made, the work follows open-source practices with clear commit messages, a structured directory, and detailed documentation. The project is planned to be uploaded to the AIORI Hackathon GitHub organization for mentor review and future collaboration.

- **Documentation includes:**
  - A step-by-step README with setup instructions.
  - RFC references linked to implemented modules.
  - Notes on Celery, AJAX, and monitoring setup for reproducibility.

This groundwork ensures that the project can be easily extended or contributed to by other teams in the AIORI network.

- **Collaboration with IETF WGs**

The project was developed by following the technical guidelines defined in several IETF RFCs related to web protocols and monitoring standards. While there was no direct interaction with IETF Working Groups, the work aligns closely with the goals of the following groups:

  - HTTPBIS WG – for HTTP semantics and web performance (RFC 9110 series).
  - OPSAREA WG – for operational measurement and network monitoring practices.
  - SEC WG / TLS WG – for secure communication standards (RFC 8446).
  - EMAILCORE WG – for email delivery and SMTP reliability (RFC 5321/5322).

Future phases aim to share implementation feedback and results through AIORI channels to the broader IETF community, especially WGs focusing on Internet operations, measurement, and security.

- **Impact and Future Work**

The Website Health Monitoring (WebCheck) project helped us understand how different Internet standards like HTTP, ICMP, and SMTP work together in real monitoring systems. It also gave us hands-on experience using these RFCs in a real Django-based setup.

This project can later become part of the AIORI Internet Measurement Network (AIORI-IMN) to monitor websites and analyze uptime from different regions.

In the next phase, we plan to:

  - Complete the email, SMS notifications.
  - Connect our system with the AIORI testbed for distributed latency testing.

These improvements will make WebCheck ready for use in real AIORI experiments and future collaborations with global Internet research teams.

- ## Lead Paragraph

In the AIORI-2 Hackathon, Our team explored how website uptime directly impacts internet reliability and user trust. Implementing a Website Health Monitoring framework ensures standards-based communication and alerting.

- ## Background and Motivation

Website downtime affects reliability, user trust, and overall Internet performance. The Website Health Monitor with Multi-Channel Alerts project implements several key RFCs — HTTP (RFC 9110–9114) for web availability checks, ICMP (RFC 792) for reachability, TLS (RFC 8446) for secure communication, and SMTP (RFC 5321/5322) for alert delivery.

These standards together enable automated detection of website outages and immediate user notifications through multiple channels. This operationally supports Internet resilience and observability, helping administrators identify downtime faster and maintain service continuity across networks — a goal aligned with the AIORI Internet Measurement framework.

- ## Technical Implementation

1. Setup and Tools

- Operating System: Windows 11 (Development Environment)
- Framework: Django 5.0 (Python-based web framework)
- Programming Language: Python 3.12
- Database: SQLite (default Django database)
- Task Scheduler: Celery 5.4 with Redis (for background monitoring tasks)
- Frontend Tools: HTML5, Bootstrap 5, AJAX, Chart.js
- Monitoring Libraries: requests, socket, ssl, time, subprocess
- Email / SMS Setup: Django SMTP backend, Twilio (test mode)
- Version Control: Git (Local repository under AIORI2-Project12/FRONT/src)
- Testing Tools: Local ping commands, HTTP response validation, and console logs

2. Implementation Steps

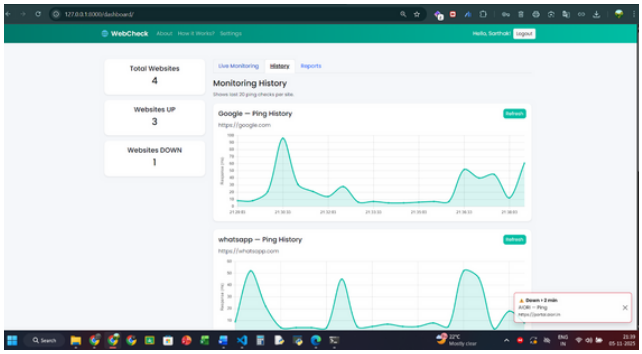
- Initialized Django Project Environment – Created the project structure (src) and set up base templates for homepage, about, and how-it-works pages.
- Developed Website Check Function – Implemented HTTP-based health checks using Python's requests library following RFC 9110/9112 for status validation.
- Added AJAX Integration – Used JSON (RFC 8259) responses to update the webpage dynamically without reload when checking site status.
- Configured Celery for Background Monitoring – Integrated Celery with Redis to perform automated periodic checks for uptime and downtime detection.
- Designed Dashboard Interface – Built a responsive Bootstrap 5 dashboard and embedded Chart.js graphs for visualizing uptime and latency trends.

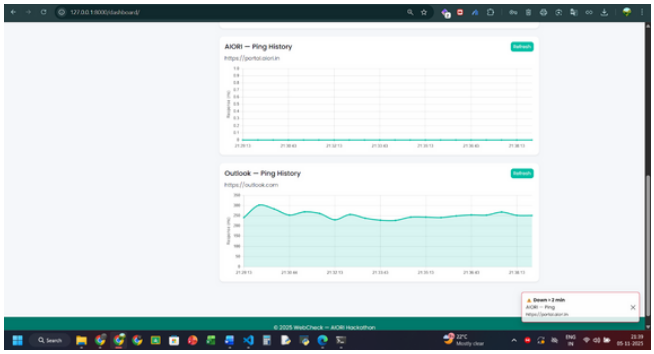
3. Challenges Faced

- Celery Interval Accuracy:
- The Celery task scheduler sometimes executed checks more frequently than the user-defined interval (e.g., 6–7 seconds instead of 15 seconds). This required tuning broker settings and understanding Celery's timing behavior beyond the basic documentation.
- Notification Integration:
- Although the models for in-app and email/SMS notifications were created, connecting them to live events and ensuring they triggered correctly during downtime proved challenging due to asynchronous task handling.
- RFC Mapping and Implementation:
- Applying HTTP (RFC 9110) and ICMP (RFC 792) semantics to real-time monitoring required simplifying protocol behaviors within Django's request model, as raw ICMP operations are restricted in user-space.
- Testing Environment Limitations:
- Since the AIORI testbed access was simulated locally, we could not perform large-scale distributed monitoring or latency comparison across multiple nodes.

- **Results and Observations**
  Include key metrics, results, and graphs. Example table below:

Test	Metric	Observation	Note
Website Status Check	HTTP Response (200 / 404 / 500)	Correctly identified UP/DOWN status of target URLs	Based on **RFC 9110** HTTP semantics
AJAX Update	Poll Interval: 5–10 sec	Real-time frontend updates without reload	JSON-based responses under **RFC 8259**
Celery Background Check	Interval Set: 15 sec	Tasks executed but sometimes repeated every 6–7 sec	Needs broker configuration tuning
Ping (ICMP) Simulation	Latency 120–300 ms	Average response time on local test URLs	Simulated **AIORI anchor** environment
Dashboard Graph	Chart.js rendering	Displayed demo data successfully	Dynamic data binding in next phase






```
[2025-11-05 21:44:14,029: WARNING/MainProcess]          Website Monitoring — 2025-11-05 21:44:03 IST

  Website    URL                        Response (ms)   Status   Checked At (IST)
  ---------  -------------------------  --------------  -------  -----------------
  Google     https://google.com                  24.00   UP       21:44:03
  whatsapp   https://whatsapp.com                30.00   UP       21:44:03
  AIORI      https://portal.aiori.in              0.00   DOWN     21:44:03
  Outlook    https://outlook.com                212.00   UP       21:44:03
  Outlook    https://outlook.com                305.00   UP       21:44:03

[2025-11-05 21:44:14,030: INFO/MainProcess] ✅ Checked 5 sites at 21:44:03 IST
[2025-11-05 21:44:14,031: INFO/MainProcess] Task monitor.tasks.run_all_monitors[7d4485a7-5213-4bd5-ba50-68d0fade97b7] succeeded in 10.9932
```

- **Lessons Learned**
  - Understanding how different RFCs (HTTP, ICMP, SMTP) interact in real-world systems is essential for building reliable monitoring tools.
  - Implementing Celery scheduling highlighted the importance of timing precision and asynchronous task management in automation.
  - Testing and debugging AJAX-based checks improved our knowledge of real-time web communication using JSON (RFC 8259).
  - We learned that small configuration issues (like Celery intervals or SMTP credentials) can significantly affect system reliability.
  - Collaboration through AIORI's sprint process reflected how IETF-style teamwork and documentation operate in real research environments.
- **Open Source and Community Contributions**

Project	Contribution	Status	Link
WebCheck	Django-based website monitoring system implementing HTTP (RFC 9110), ICMP (RFC 792), and SMTP (RFC 5321) standards	Local Repository	https://github.com/Starc6254/DNinjaS-Website-Health-Monitoring--AIORI-2-

- **Future Work**
  - **Integrate with AIORI-IMN Framework:** Connect WebCheck's monitoring engine with the AIORI Internet Measurement Network to collect distributed uptime and latency data across nodes.
  - **Enable Multi-Channel Alerts:** Complete implementation of email, and SMS notifications with verified SMTP and Twilio integration.
  - **Add Live Analytics:** Link real monitoring data to the dashboard for live uptime graphs, downtime history, and performance insights.
  - **Enhance Security & Protocol Coverage:** Extend HTTPS and TLS (RFC 8446) validation, add retry logic, and include WebSocket-based live push updates.
  - **Open Source Release:** Prepare the codebase for public release under AIORI's GitHub, inviting collaboration and contributions from the research community.

AIORI-2: Reporting and Standards Mapping

Team Name	Institution	Project Title	Focus Area
DNinjaS	International Institute of Information Technology, Pune	Website Health Monitor with Multi-Channel Alerts	Website Health Monitoring / HTTP – ICMP – TLS – SMTP Integration

Date: 5/11/2025

1. Standards Reference

RFC / Draft No.	Title / Area	Lifecycle Stage	How This Work Relates
RFC 9110–9114	HTTP Semantics and HTTP/1.1–3 Protocols	Internet Standard	Implements website health checks and response validation using HTTP status codes
RFC 792 / RFC 4443	Internet Control Message Protocol (ICMPv4/v6)	Internet Standard	Used for basic ping and reachability testing to detect downtime.
RFC 8446	The Transport Layer Security (TLS) Protocol v1.3	Internet Standard	Used for HTTPS validation and secure connection checks during monitoring
RFC 8259	The JavaScript Object Notation (JSON) Data Interchange Format	Internet Standard	Used in AJAX responses for real-time frontend communication without page

2. Impact on Standards Development

Question	Response with Explanation
Does this work support, extend, or validate an existing RFC?	Yes. The project validates several existing RFCs including RFC 9110–9114 (HTTP semantics and status codes), RFC 792 (ICMP reachability), and RFC 5321/5322 (SMTP email delivery) by implementing them in a real monitoring system that checks website health and notifies users when
Could it influence a new Internet-Draft or update sections of an RFC?	Potentially, yes. The monitoring logic and alert workflow could contribute to discussions around **Internet service reliability metrics** or **automated notification standards** in IETF operations or measurement working groups.
Any feedback or data shared with IETF WG mailing lists (e.g., DNSOP, SIDROPS, DPRIVE, QUIC)?	No direct mailing list participation yet. However, results are aligned with the interests of **OPSAREA**, **HTTPBIS**, and **EMAILCORE** WGs, and can later be shared through AIORI's collaborative channels for feedback.
Planned next step (e.g., share measurement dataset / open PR / draft text).	The next step is to prepare the **monitoring dataset and logs** for sharing through the **AIORI Internet Measurement Platform (AIORI-IMN)** and eventually contribute the implementation details to AIORI's open-source repository for broader research use.

- **References**

  - RFC 9110–9114 – HTTP Semantics and HTTP/1.1–3 Protocols
  - RFC 792 / RFC 4443 – Internet Control Message Protocol (ICMPv4/v6)
  - RFC 8259 – The JavaScript Object Notation (JSON) Data Interchange Format
  - RFC 8446 – The Transport Layer Security (TLS) Protocol Version 1.3
  - RFC 5321 / RFC 5322 / RFC 8314 – Simple Mail Transfer Protocol (SMTP) and Message Format
  - RFC 8030 / RFC 8292 / RFC 6455 – Web Push and WebSocket Protocols
  - AIORI Testbed Documentation: https://aiori.in/testbed
  - IETF HTTPBIS Working Group: https://datatracker.ietf.org/wg/httpbis/
  - IETF OPSAREA Working Group: https://datatracker.ietf.org/wg/opsarea/
  - https://doi.org/10.1093/nar/gkab396 Aviator a web service for monitoring availability of web services
  - JETIR2412359 http://www.jetir.org/
  - https://doi.org/10.22214/ijraset.2024.59855

- **About the Authors**

DNinjaS represents International Institute of Information Technology (Pune), part of the AIORI-2 Hackathon (Nov 2025). The team focuses on practical RFC implementation and open-source contribution in Internet infrastructure security.

  - Sarthak Dhamale: This project deepened my understanding of RFC-based monitoring and how real-time alerts improve Internet reliability.
  - Ayush Nakhale: I learned how protocols like HTTP and ICMP work together in detecting downtime and ensuring seamless service availability.

- **Contact**

  - Lead Author: Sarthak Dhamale
    - Email: sarthak.s.dhamale@gmail.com
  - Mentor: Prof. Bhagyashri Tanaji Thorat