tunggu.online

Requirements Document

29 July 2025

STAKEHOLDERS

Internal Stakeholders

Product Owner / Project Manager, Engineering Team (Frontend, Backend), UI/UX Designer, DevOps / Cloud Engineer, Marketing / Sales, Customer Support.

External Stakeholders

Client Website Owners, End Users (Pengunjung Website Client), Cloud Provider (Vercel, NeonDB), Regulatory / Compliance Entities

OVERVIEW

Tunggu Monitoring is a **web monitoring platform** similar to Google Analytics or Hotjar. It provides:

- A monitor.js snippet to be embedded in client websites.
- Collection of visitor activity such as page views, clicks, scroll depth, and click coordinates (for heatmaps).
- A dashboard built with **Next.js App Router** that includes authentication, site management, heatmap visualization, and analytics.

OBJECTIVES

- Deliver **visitor insights**: page views, click behavior, scroll depth.
- Provide **click heatmaps** to improve website UX.
- Enable **multi-user** and **multi-site** support (each user can register multiple sites).

SCOPE

Phase 1 (MVP)

1. **Tracking Snippet**
    - monitor.js script for embedding.
    - Events captured: pageview, click (with X,Y coordinates), and scroll (25%, 50%, 75%, 100%).
2. **Authentication & User Management**
    - Login using NextAuth (Credentials Provider).
    - Registration endpoint & page.
3. **Site Management**
    - Add new site (name & domain).
    - List all user-owned sites.
4. **Dashboard**
    - Overview page (list of sites).
    - Heatmap viewer for click visualization.
    - Analytics charts (daily visitors).
5. **Database**
    - **User**: id, email, passwordHash.
    - **Site**: id, name, domain, userId.
    - **Event**: id, siteId, type, url, referrer, ua, ip, screenWidth, screenHeight, extra(JSON), createdAt.

Phase 2

- Session & unique visitor tracking.
- Real-time analytics (WebSocket).
- User roles (admin vs regular user).
- Date range filters for heatmaps.

ARCHITECTURE

Tech Stack

- **Frontend**: Next.js 14 (App Router), TailwindCSS 3.
- **Authentication**: NextAuth with Credentials Provider.
- **Database**: NeonDB (PostgreSQL) using Prisma ORM.
- **Hosting**: Vercel (App + API), NeonDB (DB).
- **Analytics Script**: monitor.js served via /public.

High-Level Flow

1. Client websites embed:

```html
<script>
  window.TungguAnalytics = { siteId: "SITE_ID" };
</script>
<script async src="https://tunggu.online/monitor.js"></script>
```

2. monitor.js captures user events and posts them to /api/event.
3. The backend saves data in the Event table.
4. Dashboard displays:
    - List of sites.
    - Click heatmaps (via heatmap.js).
    - Analytics charts (via react-chartjs-2).

(EARLY) STRUCTURE

```
tunggu-monitoring/
├── app/
│   ├── layout.tsx              # Root layout
│   ├── page.tsx                # Landing page
│   ├── dashboard/
│   │   ├── layout.tsx          # Dashboard layout (sidebar + topbar)
│   │   ├── page.tsx            # Sites list & add form
│   │   ├── heatmap/page.tsx    # Heatmap viewer page
│   │   └── analytics/page.tsx  # Analytics charts page
│   └── api/
│       ├── auth/[...nextauth]/route.ts # NextAuth config
│       ├── site/route.ts       # API for site CRUD
│       └── event/route.ts      # API for event tracking
├── components/
│   ├── Sidebar.tsx             # Sidebar navigation
│   ├── HeatmapViewer.tsx       # Heatmap visualization component
│   └── Layout.tsx              # Layout wrapper
├── lib/
│   ├── prisma.ts               # Prisma client instance
│   └── auth.ts                 # Auth helper
├── prisma/
│   └── schema.prisma           # Database schema
├── public/
│   └── monitor.js              # Tracking snippet
├── types/
│   └── next-auth.d.ts          # Session type extension
├── styles/
│   └── globals.css
├── tailwind.config.js
├── postcss.config.js
└── next.config.js
```

DATABASE SCHEMA

```
model User {
  id            String   @id @default(cuid())
  email         String   @unique
  passwordHash  String
  sites         Site[]
  createdAt     DateTime @default(now())
}

model Site {
  id        String   @id @default(cuid())
  name      String
  domain    String
  user      User     @relation(fields: [userId], references: [id])
  userId    String
  events    Event[]
  createdAt DateTime @default(now())
}

model Event {
  id            String   @id @default(cuid())
  site          Site     @relation(fields: [siteId], references: [id])
  siteId        String
  type          String
  url           String?
  referrer      String?
  ip            String?
  ua            String?
  screenWidth   Int?
  screenHeight  Int?
  extra         Json?
  createdAt     DateTime @default(now())
}
```

API DESIGN

/api/event (POST)

Request Body:

```
{
  "siteId": "uuid",
  "type": "pageview|click|scroll",
  "url": "https://example.com/page",
  "referrer": "https://google.com",
  "ua": "Mozilla/5.0 ...",
  "screen_width": 1920,
  "screen_height": 1080,
  "extra": { "x": 120, "y": 240 }
}
```

Response: { "success": true }

/api/site (POST)

Request Body:

```
{ "name": "Example Site", "domain": "example.com" }
```

Response: Site object JSON.

UI / UX

- **Login Page**: Email & Password form.
- **Dashboard Layout**: Sidebar (My Sites, Heatmap, Analytics) + Topbar.
- **Heatmap Page**: Heatmap overlay using heatmap.js.
- **Analytics Page**: Visitor bar chart using react-chartjs-2.

DEPLOYMENT

Deploy to **Vercel**:

- Environment variables: DATABASE_URL, NEXTAUTH_SECRET.

Deploy database to **NeonDB**:

- Run npx prisma migrate deploy.

DEV STEPS

- Add login and register pages.
- Add date filters for heatmap & analytics.
- Implement unique visitor session tracking.