Issue Tracker

Web Application – Documentation

Live Demo (Frontend)	https://issue-tracker-steel-two.vercel.app
Backend API Base	https://issue-tracker-backend-hpyn.onrender.com
Test Credentials	Email: user@user.com Password: user1234
Date	January 07, 2026

This document explains the full implementation of the Issue Tracker project (frontend + backend), including setup instructions, key features, security controls, and deployment details.

Overview

Issue Tracker is a full-stack web application used to create, view, update, and delete issues. It includes authentication, filtering, searching, pagination, and export features.

Live URLs

Frontend: https://issue-tracker-steel-two.vercel.app
Backend: https://issue-tracker-backend-hpyn.onrender.com

Test Account

Email: user@user.com
Password: user1234

Features

• **Authentication**: register, login, logout, refresh session.
• **Issue CRUD**: create, view list, view details, edit, delete (with confirmation).
• **Search + filters**: status / priority + keyword search; optimized requests.
• Pagination for issue list.
• Export issue list to CSV or JSON.
• **Monitoring:** /metrics (Prometheus format) and /api/health.

Tech Stack

Frontend	React (Vite) + TypeScript, Tailwind CSS, Redux Toolkit Query
Backend	Node.js + Express (TypeScript)
Database	MongoDB (Mongoose)
Auth	JWT access + refresh tokens stored in HTTP-only cookies
Deployment	Vercel (frontend) + Render (backend)

System Architecture

High-level flow:

1) User opens the frontend in the browser (Vercel).

2) Frontend calls the backend REST API (Render) with credentials included.

3) Backend validates auth cookies (JWT) and performs CRUD on MongoDB.

4) Backend responds with JSON; frontend updates UI using RTK Query caching.

Mobile Safari note

Some iPhone browsers are strict with cross-site cookies. If needed, configure the frontend to proxy API requests through the same domain (Vercel rewrite from /api/* to the Render backend). This makes cookies first-party and improves login stability on iOS.

Deployment

- **Backend (Render)**

Root directory: server
Build command: npm ci --include=dev && npm run build
Start command: npm run start
Health check path: /api/health

- **Frontend (Vercel)**

Configure SPA rewrite so direct URLs like /login work after refresh. Optional: add an API proxy rewrite (/api/* to backend) to improve cookie behavior on iOS.

Security Controls

- CORS allowlist with credentials enabled (only trusted frontend origins).
- HTTP-only cookies for tokens (not readable by JavaScript).
- CSRF guard for state-changing requests based on Origin allowlist.
- Helmet for common secure HTTP headers.
- Rate limiting to reduce brute-force and abuse.
- HPP + request sanitization for safer input handling.
- Log redaction for authorization and cookies

API Summary

Base path: /api/v1

Area	Endpoints (examples)
Auth	POST /auth/register, POST /auth/login, GET /auth/me, POST /auth/logout, POST /auth/refresh
Issues	GET /issues, POST /issues, GET /issues/:id, PATCH /issues/:id, DELETE /issues/:id
Export	GET /issues/export?format=csv, GET /issues/export?format=json
Health/Metrics	GET /health (also /api/health), GET /metrics

Troubleshooting

- If login works on desktop but not on iPhone, use a Vercel API rewrite proxy to make cookies first-party.

- If CORS blocks requests, confirm CLIENT_ORIGIN includes https:// and matches the Vercel URL Exactly.

- If Render shows 'Application loading', ensure the server listens on process.env.PORT and the health endpoint responds.