Onit — Ticketing System Documentation

Overview

- Project: Onit — HTML/JS ticketing system with a Node.js backend.
- Purpose: Submit, route, view and manage support tickets across departments (Branch, ICT, Finance, etc.) with demo/local fallback and optional public tunneling for remote testing.

Languages & Tech Stack

- Frontend: HTML (multiple pages), CSS (`modern-ui.css`, Tailwind CDN usage in pages), JavaScript (inline scripts and `script.js`).
- Backend: Node.js + Express (`server.js`, `minimal-server.js`) providing REST endpoints.
- Data / Formats: JSON files (`tickets.json`, `users.json`, `asset-register.json`) and browser `localStorage`.
- Extras: Batch/Windows helper (`start-system.bat`), seed scripts (`seed-users-standalone.js`, `test-seed.js`), README/MD docs.

Global Concepts

- Primary storage: Unified `tickets` in browser `localStorage` (with support for legacy per-department keys like `branchTickets`, `ictTickets`, `financeTickets`).
- Ticket schema (standardized fields): `id`, `department`, `created_by`, `created_by_email`, `assigned_to`, `timestamp`, `status`, `priority`, `history`, `resolutionNotes`, plus other metadata.
- Auto-assignment: On submission, code fetches department users, counts open tickets per agent, and assigns to the least-loaded agent.
- API endpoints (server): `/api/users`, `/api/tickets`, `/api/health` used by the frontend to fetch users and tickets and to health-check the service.
- Demo fallback: Pages seed demo users into `localStorage` (e.g., `ictUsers`) and use that when the API is unreachable.
- Tunneling for remote tests: loca.lt used to expose local server; note loca.lt may show an access gate requiring a password (host IP) or require reserving a subdomain.

Pages & Department Functionality

- `index.html`

  - Landing / navigation hub linking to department portals and system pages.

- `submit.html`

  - Master ticket submission page used to create tickets.
  - Collects requester info, department, priority, description.
  - Saves to unified `tickets` store and triggers auto-assignment logic where implemented.

- `tickets.html`

  - Ticket list / shared viewer showing tickets filtered by department or global view.
  - Uses `tickets` from `localStorage` or the backend when available.

- `branch.html` (Branch department)

  - Branch portal: shows tickets routed to Branch (accepts legacy `toDept` fallback).
  - Allows branch staff to view ticket details, update status and resolution notes.
  - Integrates with auto-assignment logic when new branch tickets are submitted.

- `ict.html` (ICT department)

  - ICT portal and login/select UI: selects ICT user (login via a select populated from `/api/users` or `localStorage` fallback).
  - Displays ICT tickets, allows claim/assignment actions, updates ticket status and history.
  - Contains logic to seed demo ICT users and a function which first fills the select from fallback, then

attempts the API.
  - Recent fixes addressed a stuck "Loading users..." state by ensuring local fallback population occurs immediately and fixed a JavaScript syntax error that previously blocked execution.

- `finance.html` (Finance department)
  - Finance portal: shows finance tickets and accepts submissions routed to Finance.
  - Supports viewing and updating tickets (status, assigned agent).

- `expense-claim.html`
  - Expense claims UI for creating and tracking claim workflows.
  - Integrates with ticketing and approval logic and stores claim metadata.

- `leave-management.html`
  - Leave request UI — submit and view leave-type tickets/requests.

- `admin.html`
  - Administrative interface for system-level actions (user management, seeding demo data, system checks).

- `system.html`, `system-clean.html`, `change-password.html`
  - System utilities: maintenance, cleanup, password change UI and system-level controls.

User & Agent Management

- `users.json` & seed scripts store demo user records; seeding scripts populate the backend or `localStorage` for development and testing.
- Department pages populate user selects (e.g., `#ictUserEmail`) using email as visible text (recent change).
- Assignment uses user identifiers; storing `assigned_to` as email is recommended for deterministic matching while UI can show friendly names.

Backend Behavior

- `server.js` and `minimal-server.js` host REST endpoints used by the frontend.
- `/api/users` returns user lists which frontend filters per department.
- `/api/tickets` supports GET/POST for ticket persistence when server is running.
- Frontend falls back to `localStorage` if the server is unreachable; API URL construction uses the page origin or `http://localhost:3003/api/...` fallback.

Data Compatibility & Known Issues

- Legacy keys: Some pages historically used `toDept` or per-department ticket stores. The codebase aims to read both legacy keys and the unified `tickets`.
- Assignment representation: Mixed uses of display name vs email for `assigned_to` exist; the recommended canonical format is email while presenting names in the UI.
- Timing/race issues: Pages that attempted to populate UI before demo seed ran could show "Loading users..." — addressed in recent `ict.html` fixes by populating from fallback first.
- Tunneling gate: Using loca.lt may present a gate requiring the host IP as a password; alternatives include reserving a subdomain or using ngrok.

Dev / Test Utilities

- Seed scripts: `seed-users-standalone.js`, `test-seed.js` create demo users/tickets for development.
- Server control: `start-system.bat` and `node server.js` are used to start the backend locally.
- Debugging aids: Console logs and temporary on-page debug elements were used during fixes and were removed post-verification.

Files of Interest

- `ict.html`, `branch.html`, `finance.html` — department portals with recent edits.
- `submit.html` — centralized submission logic and ticket normalization.
- `server.js`, `minimal-server.js` — backend API implementations.
- `tickets.json`, `users.json` — sample data for developer usage.
- `script.js` — shared client-side helpers used across pages.
- `modern-ui.css` — project styling; Tailwind CDN used on some pages.

Operational Notes & Recommendations

- Canonicalize `assigned_to` as email and update submission/portal views to store emails while displaying "Full Name <email>" where helpful.
- Run an end-to-end test: submit a ticket from `submit.html` and verify it appears in `branch.html`, `ict.html`, and `finance.html` (validate both `localStorage` and `/api/tickets` paths).
- Choose a tunnel approach: if loca.lt gate is undesirable, consider ngrok (requires an ngrok account and authtoken) or recreate a loca.lt reserved subdomain and verify from remote testers.
- Add `API.md`: document endpoints and payload formats for stable frontend/backend integration.
- Add simple integration checks: scripts that POST a ticket and GET department views to assert end-to-end behavior.

End of Documentation