Assignment: Mini "Buyer Lead Intake" App

Goal: Build a small app to **capture, list, and manage buyer leads** with sane validation, search/filter, and CSV import/export.

Deadline: 15th September 2025 by 13:00 IST

Deliverables: Public GitHub repo (+ optional Vercel deploy) and a short README.

Stack (must)

- Next.js (App Router) + TypeScript
- DB: Postgres/Supabase/SQLite with Drizzle or Prisma + **migrations**
- Zod for validation
- Auth: simple magic link or demo login
- Git with meaningful commits

Data Model (must)

buyers (aka leads)

- `id` (uuid)
- `fullName` (string, 2–80)
- `email` (email, optional)
- `phone` (string, 10–15; required)
- `city` (enum: `Chandigarh|Mohali|Zirakpur|Panchkula|Other`)
- `propertyType` (enum: `Apartment|Villa|Plot|Office|Retail`)
- `bhk` (enum: `1|2|3|4|Studio`, optional if non-residential)
- `purpose` (enum: `Buy|Rent`)
- `budgetMin` (int, INR; optional)
- `budgetMax` (int, INR; optional; must be ≥ `budgetMin` if both present)
- `timeline` (enum: `0-3m|3-6m|>6m|Exploring`)
- `source` (enum: `Website|Referral|Walk-in|Call|Other`)
- `status` (enum: `New|Qualified|Contacted|Visited|Negotiation|Converted|Dropped`) – default `New`
- `notes` (text, optional, ≤ 1,000 chars)
- `tags` (string[], optional)
- `ownerId` (user id)
- `updatedAt` (timestamp)

buyer_history

- `id`, `buyerId`, `changedBy`, `changedAt`, `diff` (JSON of changed fields)

Pages & Flows (must)

1) Create Lead – `/buyers/new`

Form fields (exact):

`fullName, email, phone, city, propertyType, bhk (conditional: only for Apartment/Villa), purpose, budgetMin, budgetMax, timeline, source, notes, tags[]`

Validation (client + server):

- `fullName` ≥ 2 chars
- `phone` numeric 10–15 digits
- `email` valid if provided
- `budgetMax ≥ budgetMin` when both present
- `bhk` required iff `propertyType ∈ {Apartment, Villa}`

On submit: create record, assign `ownerId` to current user, write an entry in `buyer_history`.

2) List & Search – `/buyers`

- **SSR** with real pagination (page size 10)
- **URL-synced filters:** `city`, `propertyType`, `status`, `timeline`
- **Debounced search:** by `fullName|phone|email`
- **Sort:** default `updatedAt` desc
- **Columns:** Name, Phone, City, PropertyType, Budget (min–max), Timeline, Status, UpdatedAt
- Row action: **View / Edit**

3) View & Edit – `/buyers/[id]`

- Show all fields; allow edit with the same validation rules
- **Concurrency:** include `updatedAt` hidden input; if stale, reject with a friendly "Record changed, please refresh" message
- **History:** render last 5 changes from `buyer_history` (field, old → new, timestamp, user)

4) Import / Export (must)

- **CSV Import** (max 200 rows) with headers:

`fullName,email,phone,city,propertyType,bhk,purpose,budgetMin,budgetMax,timeline,source,notes,tags,status`

    - Validate per row; show a table of errors (row # + message)
    - Insert only valid rows in a **transaction**
    - Unknown enums → error
- **CSV Export** of **current filtered list** (respect filters/search/sort)

Ownership & Auth (must)

- Anyone logged in can **read** all buyers
- Users can **edit/delete only their own** (`ownerId`)
- (If you add an `admin` role, admin can edit all—optional)

Nice-to-haves (pick any 2–3)

- Tag chips with typeahead
- Status quick-actions (dropdown in table)
- Basic full-text search on `fullName,email,notes`
- Optimistic edit with rollback
- File upload for a single `attachmentUrl` (optional doc)

Quality Bar (must)

- 1 unit test (e.g., CSV row validator or budget validator)
- Simple rate limit on create/update (per user/IP)
- Error boundary + empty state
- Accessibility basics: labels, keyboard focus, form errors announced

Scoring (100)

- **Correctness & UX (30):** CRUD, filters/search, URL sync, helpful errors
- **Code Quality (20):** structure, typing, migrations, commits
- **Validation & Safety (15):** Zod both sides, ownership checks, rate limit
- **Data & SSR (15):** real pagination/sorting/filter on server
- **Import/Export (10):** transactional import + row errors, filtered export
- **Polish/Extras (10):** tests, a11y, any nice-to-haves

README (what to include)

- Setup (env, migrate/seed commands), how to run locally
- Design notes: where validation lives, SSR vs client, ownership enforcement
- What's done vs skipped (and why)