SHUBHAM ADVERTISE – TUESDAY DELIVERABLES SPEC

Objective
Provide a working demo build by Tuesday that showcases practical functionality for multi-user access, hoarding master data, rent management, and rent reminders (dashboard + notifications + email). This document defines the minimum scope the developer must complete.

Deadline
• Code freeze for this scope: Tuesday (EOD), ready for client demo.

----
1. Multi-User Login & Account Setup

Goal:
Different team members must be able to log in with their own credentials and see only what is relevant to their role.

Requirements:
1.1 Implement a working authentication flow
- Login via email/username + password (any standard approach is fine for now).
- Proper session handling (user remains logged in until logout or token expiry).

1.2 User roles to support at minimum:
- Owner
- Manager
- Sales
- Designer
- Fitter / Execution (person who fits/installs the hoarding)

1.3 On successful login:
- Redirect each user to a basic dashboard/home screen (role-specific content is enough even if simple).
- Show the logged-in user's name and role somewhere in the header/top bar.

1.4 Seed/Test data:
- Create at least 3–4 test user accounts with different roles so that we can demonstrate logging in as different users during the client demo.

Acceptance criteria:
- I can log in as "Owner" and see Owner options.
- I can log in as "Sales" and see a simpler view with limited options.
- Logging out and logging in as another role works correctly.

----
2. Hoarding Master Database (Main Data Layer)

Goal:
Have a working hoarding master where real hoarding data is stored, editable, and visible in the app.

Requirements:
2.1 Database design:
- Create a "Hoardings" master table with at least these fields:
  – hoarding_id (unique identifier)
  – location / area
  – road / landmark

    – size (e.g., “20 x 10”)
    – side / position (LHS / RHS / Upper / Lower etc.)
    – illumination (Lit / Non-lit / LED etc.)
    – status (Available / Booked / On Rent / Expired etc.)
    – party_type (Government / Private / Friend) – can also be kept in rent table if you prefer.

2.2 UI/Screen:
- Provide a “Hoardings Master” screen that shows data in a table/grid.
- Basic actions:
  – View list of all hoardings.
  – Basic filters/search by: area, size, status (even simple search is fine for demo).
- Each hoarding row must have a “Rent” action/button (see Section 4).

2.3 Seed data:
- Insert realistic sample data for multiple hoardings (from our sheet / dummy values) so the table looks populated during the demo.

Acceptance criteria:
- From the menu/dashboard, I can open “Hoardings Master” and see a populated table.
- I can click on an individual hoarding’s “Rent” button to manage rent details.

----
3. Role-Based Access Control (RBAC)

Goal:
Ensure different roles have different levels of access and visibility.

Requirements:
3.1 Role definitions (minimum):
- Owner – full access to all features, settings, and data.
- Manager – access to hoardings, rent details, dashboards, and team overview (except high-level admin settings).
- Sales – access only to hoarding list, enquiries/leads, and booking-related features.
- Designer – access only to design/creative-related sections (can be a placeholder section for now).
- Fitter / Execution – access only to their assigned hoarding jobs/tasks (can be simple list now).

3.2 Owner capabilities:
- Owner can view all users and assign roles to them.
- Owner can create new roles (custom name) – basic CRUD on roles is enough, even if permissions for custom roles are minimal for now.

3.3 UI behaviour:
- Non-owners must not see admin/role management menu items.
- If a user tries to access a restricted URL/section, show “Access Denied” or redirect them back to their dashboard.

Acceptance criteria:
- Logged-in Owner can see “User & Role Management” section.
- Logged-in Sales/Designer/Fitter cannot see or access that section.
- Role-based visibility is clearly different between Owner and other roles in the demo.

----
4. Rent Management (Per Hoarding)

Context:
Rent is paid to three types of parties: Government, Private, and Friend.

Goal:
From the hoarding master, we should be able to manage rent information per hoarding and track due dates.

Requirements:
4.1 “Rent” button in Hoarding Master:
- In each hoarding row, add a “Rent” button.
- On click, redirect to a dedicated “Rent Details” page for that hoarding.

4.2 Rent Details Page fields:
For each hoarding, the Rent Details page must have:

- Party Type:
  – Dropdown: Government / Private / Friend
- Rent Amount:
  – Numeric input, represents how much rent is there (e.g., 1000 per month).
- Increment Year:
  – Year in which rent increment is applicable (e.g., 2026).
  – Increment rule: rent increased by 10%.
- Payment Mode:
  – Dropdown with options: Yearly, Half-Yearly, Quarterly, Monthly.
- Last Payment Date:
  – Date picker – when last payment was done.
- Next Due Date:
  – Auto-calculated based on Last Payment Date + Payment Mode.

4.3 Rent increment logic:
- Basic version: when the current date crosses the Increment Year, system calculates new rent as:

```
new_rent = old_rent + (old_rent * 0.10)
```

- For Tuesday demo, it is acceptable if increment calculation is triggered manually on save or via a button (e.g., “Recalculate Rent”), as long as the logic works correctly.

4.4 Data persistence:
- All rent details must be stored in the database linked to the specific hoarding.
- When revisiting a hoarding’s rent page, previously saved values must appear.

Acceptance criteria:
- From Hoarding Master, click “Rent” for a hoarding → go to Rent Details page.
- Fill rent info, save, and see values saved.
- Next Due Date is correctly calculated based on last payment date + payment mode.
- When Increment Year is reached or recalculated, rent shows as increased by 10%.

----
5. Dashboard – Rent Overview & Reminders

Goal:
Provide a high-level summary of rent and upcoming due dates, along with automated reminders.

5.1 Rent Summary (Dashboard Cards)
For Owner and Manager dashboards, show at least:

• Total Hoardings on Rent
  – Example: “20 Hoardings on Rent”
• Total Rent Amount
  – Example: “■20,000 Total Rent”
  – For demo, assume each hoarding’s rent is 1000 and compute sum accordingly.
  – This can be per-month equivalent or based on payment mode; keep it consistent and display clearly.

5.2 Due Date List
• Add a dashboard section like “Upcoming Rent Due”.
• Show a list/table with:
  – Hoarding ID / Location
  – Party Type (Government / Private / Friend)
  – Rent Amount
  – Next Due Date
• Sort by closest Next Due Date first.

5.3 Email Reminder to Owner
• Implement an automatic email reminder mechanism for upcoming due dates.
• Behaviour (for demo):
  – For all hoardings where Next Due Date is within the next X days (e.g., 7 days), send an email to the Owner.
  – Email content should contain:
    · Hoarding ID / Location
    · Party Type
    · Rent Amount
    · Exact Due Date
• For Tuesday demo, it is acceptable if this is triggered by:
  – A manual button on the dashboard like “Send Rent Reminders Now”, OR
  – A simple scheduled task (if easy to configure).
• At least one successful email must be demonstrated.

5.4 In-App Notification / Alert
• Show a visual reminder inside the app before due dates.
• Examples (any one is fine for demo):
  – A notification badge on the dashboard: “3 hoardings have rent due in next 7 days”.
  – A highlighted banner / alerts section listing upcoming dues.
• Clicking the notification/reminder should open the “Upcoming Rent Due” list/table.

Acceptance criteria:
• Owner dashboard shows total hoardings on rent + total rent amount.
• Owner/Manager can see a list of upcoming rent due entries.
• Triggering the reminder action sends an email to the Owner for at least one sample hoarding.
• An in-app notification/badge/alert is visible when due dates are approaching.

----
6. General Notes & Expectations

6.1 Priority for Tuesday demo:
• End-to-end flow must be practically working:
  – Login as Owner → open Hoardings Master → open Rent page for a hoarding → save rent details → see due date → see dashboard summary → show upcoming due list → trigger email reminder/notification.
• UI can be basic, but it must be clean and understandable.

6.2 Code quality:

- Use clear naming for models, controllers, and database tables.
- Add minimal comments where logic is non-obvious (especially for rent calculation and due-date logic).

6.3 Communication:

- If any item cannot be completed by Tuesday, update with:
  - What is done
  - What is partially done
  - What is pending

  so we can plan the client demo accordingly.

This scope is the minimum required for a “working condition” demo to the client. Further UI/UX polish and advanced features can be added in the next phase once this base is stable.