System Overview

- **Project Title:** Improved Clinic Management System for DRMed Clinic and Laboratory
- **Client:** DRMed Clinic and Laboratory
- **Objective:** To streamline clinic operations by providing an improved system for managing patient records, laboratory requests, laboratory results, and test components/packages efficiently. The system includes finalized features for user roles, data storage, template creation, result input/release, and reporting, based on user-centric design principles.

Actors and Roles

Based on the finalized Use Cases and Non-Functional Requirements:

- **Receptionist:**
  - Manages patient records (Create UC#01, Search/Retrieve UC#03, Update UC#04, View UC#14).
  - Creates Laboratory Requests (UC#02).
  - Releases completed Lab Results (handles download/status updates) (UC#08).
  - Views Laboratory Request Records (UC#05).
- **Owner:**
  - Manages Test Components (Add UC#09, Update UC#11), including creating/editing Result Templates via a form builder.
  - Manages Test Packages (Add UC#10, Update UC#12).
  - Manages Laboratory Technicians (Add UC#13, Update UC#15), including signature images.
  - Can perform Receptionist duties related to patient records (Create UC#01, Update UC#04, View UC#14).
  - Registers new users (NFR#1).
- **Laboratory Technician:**
  - Inputs new Lab Results using predefined templates (UC#06).
  - Edits previously saved (draft) Lab Results (UC#07).
  - Reviews and approves lab result values (Req#13).
  - Views Laboratory Request Records (UC#05).

Core Functional Requirements

Based on the finalized HIGH and MEDIUM priority requirements (Feb 17, 2025 doc):

- **Patient Management:**
  - Manage a central patient repository (`patient` table) accessible when needed (Req#1).
  - Allow creation and updating of patient records (Req#2, Req#31).
  - Automatically assign a unique `patient_id` (Req#3).
  - Detect potential duplicate patient records during creation and alert the user (Req#23).
  - Provide search functionality for patient records using various criteria (Req#24, Req#29).
  - Generate a preview of patient records before saving (Req#30).
- **Test/Package Management:**
  - Enable creation and management of individual test components (`test_component` table) with price and category (Req#4).
  - Provide a form builder for creating customizable test component result templates (`template_form`, `template_section`, `template_field` tables) (Req#5).
  - Allow defining constraints (e.g., ranges, max length) for fields within templates (Req#33).
  - Allow viewing of created template forms (Req#32).

    - Enable creation and management of test packages (`test_package` table) grouping multiple components (Req#6).
    - Allow previewing components within a package (Req#35).
    - Allow updating test components and their associated templates (Req#34).
    - Allow updating test packages (Req#36).
- **Lab Request Management:**
    - Store all laboratory request records (`lab_request` table) (Req#9).
    - Allow users (Receptionist) to create Laboratory Request Records, selecting patient, tests/packages (Req#10).
    - Prevent adding duplicate test components within a single lab request (especially when packages are used) (Req#25).
    - Track and display the status of each test component within a request (e.g., "Not Started", "In Progress", "Completed") (Req#11).
    - Track the overall status of the lab request (implicitly covered by component statuses and release).
    - Display requested components/packages in appropriate tabs during creation (Req#28).
    - Allow users to deselect tests/packages during creation (Req#27).
    - Automatically insert patient info when creating a request via Patient ID lookup (Req#29).
    - Generate a billing summary upon request creation, applying PWD/Senior discounts automatically (Req#26, Req#38).
    - Allow optional input of physician name (Req#40).
    - Allow selection of result delivery mode (Pick-up, Email) via checkboxes (Req#39).
    - Display all requested components within a lab request view (Req#19).
    - Display relevant patient details (Name, ID, PWD/Senior ID) when viewing a request (Req#20).
    - Sort lab requests by date on the lab technician screen (Req#41).
- **Result Management & Templates:**
    - Store result values (`result_value` table) for each requested test component (Req#15).
    - Link specific Result Templates (`template_form`) to corresponding requested test components (`request_line_item`) (Req#16).
    - Allow users (Lab Technician) to input results into these templates (Req#17).
    - Allow saving partially filled results and continuing edits later (Req#18).
    - Allow Lab Technicians/Medical Technicians to review and approve results (Req#13).
    - Allow viewing of lab results once created (even if "Not Started") (Req#44).
    - Display patient and request details when viewing/inputting results (Req#21).
    - Generate a PDF version of lab results for selected components upon request (Req#22).
    - Log the collection of laboratory results with timestamps and mode (Req#12).
    - Enable Receptionists to change digital result status to "complete" upon emailing (Req#42).
    - Enable Receptionists to change printed result status to "complete" upon pickup (Req#43).
- **Lab Technician Management:**
    - Maintain a repository of laboratory technicians (`lab_tech` table) with names, signatures, titles, roles, PRC license (Req#7).
    - Allow users (Owner) to add/delete technician details (Req#8).
    - Allow uploading PNG signature images for technicians (Req#37).
    - Link technician signatures to results they are responsible for (Req#37).
- **Auditing & Reporting:**

    - Maintain an audit log history for lab requests, tracking changes and timestamps (Req#14).

Key Non-Functional Requirements
Based on the finalized list (Feb 17, 2025 doc):

- **Security:**
    - Admin (Owner) must be able to register users (`user` table) (NFR#1).
    - Secure login/logout mechanism using unique username/password (NFR#2).
    - Role-Based User Interface (RBUI) restricting access based on role (Receptionist, Owner, Lab Technician) (NFR#3).
    - Enforce data validation (formats for dates, email, phone, numeric values) on input fields across all forms (NFR#4).
- **Performance:**
    - System must respond to user actions within 2 seconds under normal load (NFR#7).
    - Search feature must maintain 100% accuracy as database grows (NFR#8).
- **Usability:**
    - Responsive design adapting to various desktop screen resolutions and window sizes (NFR#9).
- **Data Integrity & Recovery:**
    - Automated daily backups of all critical data (requests, results, logs) to secure cloud storage (NFR#5).
    - Ability to restore data from backups (NFR#6).
    - Data validation to prevent errors and ensure accuracy (NFR#4).
    - Hide laboratory service records older than one year to keep lists current (NFR#10).
- **Reliability:**
    - Search accuracy maintained with data growth (NFR#8).
    - System should handle normal load without noticeable lag or crashes (NFR#7).

Core Workflows (Use Case Summary)
Based on the finalized Use Cases (Feb 17, 2025 doc):

- **UC#01: Create Patient Record (Actor: Receptionist/Owner):** Add a new patient, fill details, handle potential duplicates, auto-generate unique Patient ID, preview before saving.
- **UC#02: Create Lab Request Record (Actor: Receptionist):** Search existing patient, select test components/packages (handling duplicates/disabling included components), specify delivery mode (email/pickup), optionally add physician name, view billing summary with discounts, save request.
- **UC#03: Search and Retrieve Patient Record (Actor: Receptionist):** Find existing patients using various search criteria (ID, Name, etc.).
- **UC#04: Update Patient Record (Actor: Receptionist/Owner):** Select existing patient, edit their information (requires password for Owner), save changes.
- **UC#05: View Laboratory Request Record (Actor: Lab Technician/Receptionist):** Access and view details of a specific lab request, including patient info, requested items, statuses, and links to results.
- **UC#06: Input Lab Results (Actor: Lab Technician):** Access a specific test component within a request, use its associated result template, input result values, add signatures, save (draft) or submit (final) results.
- **UC#07: Edit Lab Results (Actor: Lab Technician):** Access a previously saved (draft) lab result form, modify inputs, save or submit.
- **UC#08: Release Lab Results (Actor: Receptionist):** View completed lab request, select results, download as PDF (ZIP for multiple), update collection status (email sent/picked up).

- **UC#09: Add Test Component (Actor: Owner):** Define a new test component (code, name, category, price), create its associated result template using the form builder (add sections/fields/types/fixed values).
- **UC#10: Add Package (Actor: Owner):** Create a new test package, assign a name and price, select existing test components to include.
- **UC#11: Update Test Component (Actor: Owner):** Edit details (price, category) of an existing test component, edit its associated result template via the form builder.
- **UC#12: Update Package (Actor: Owner):** Edit details (price) of an existing package, add/remove included test components.
- **UC#13: Add Lab Tech (Actor: Owner):** Create a record for a new laboratory technician, inputting details (name, title, role, license) and uploading their signature image.
- **UC#14: View Patient Record (Actor: Receptionist/Owner):** Access the detailed view of a patient, showing personal information and a history of their laboratory requests.
- **UC#15: Update Lab Tech (Actor: Owner):** Edit details of an existing lab technician (excluding name for security), possibly update signature image.

Data Model Summary
Based on the finalized ERD and Data Dictionary (Feb 17, 2025 doc):

- **Primary Entities:**
  - `user`: Stores login credentials (username, hashed password) and role (`Owner`, `Receptionist`, `Lab Technician`). PK: `username`.
  - `patient`: Stores patient demographic and contact information. PK: `patient_id`. Linked to `address`.
  - `address` (Normalized from `patient`): Stores address components (house number, street, city, etc.). PK: `address_id`. FK: `address_id` in `patient`.
  - `lab_request`: Tracks a single laboratory service request instance for a patient. PK: `request_id`. FKs: `patient_id`.
  - `collection_log`: Logs when and how results were collected/released. PK: `collection_id`. FK: `request_id`.
  - `test_component`: Defines individual laboratory tests offered. PK: `component_id`. FK: `template_id`.
  - `template_form`: Defines the structure (template) for entering results for a `test_component`. PK: `template_id`.
  - `template_section`: Defines sections within a `template_form`. PK: `section_id`. FK: `template_id`.
  - `template_field`: Defines specific input fields within a `template_section`. PK: `field_id`. FK: `section_id`.
  - `test_package`: Defines bundled packages of tests. PK: `package_id`.
  - `lab_tech`: Stores information about laboratory technicians. PK: `lab_tech_id`.
  - `lab_result`: Stores the overall result instance linked to a specific line item in a request. PK: `result_id`. FK: `line_item_id`.
  - `result_value`: Stores the actual value entered for a specific field in a lab result. PK: `result_value_id`. FKs: `result_id`, `field_id`.
  - `request_line_item`: Represents a single test or package requested within a `lab_request`. PK: `line_item_id`. FKs: `request_id`, `package_id`(nullable), `component_id`(nullable).
  - `status_log`: Tracks status changes for each `request_line_item`. PK: `status_log_id`. FK: `line_item_id`.
  - `result_review`: Tracks reviews of `result_value` by technicians. PK/FKs: `lab_tech_id`, `result_value_id`.
- **Linking/Junction Tables:**

    - `test_package_component`: Links `test_package` and `test_component` (Many-to-Many). PK/FKs: `package_id`, `component_id`.
    - `request_component`: (Implicitly handled by `request_line_item` when `component_id` is not null). Links individual components directly to a `lab_request`.
    - `request_package`: (Implicitly handled by `request_line_item` when `package_id` is not null). Links packages to a `lab_request`.
- **Key Relationships (via PK/FK):**
    - `patient` -> `lab_request` (One-to-Many)
    - `lab_request` -> `request_line_item` (One-to-Many)
    - `request_line_item` -> `lab_result` (One-to-One, potentially)
    - `lab_result` -> `result_value` (One-to-Many)
    - `template_field` -> `result_value` (One-to-Many)
    - `test_component` -> `template_form` (One-to-One, conceptually)
    - `template_form` -> `template_section` (One-to-Many)
    - `template_section` -> `template_field` (One-to-Many)
    - `test_package` <-> `test_component` (Many-to-Many via `test_package_component`)
    - `lab_tech` -> `result_review` <- `result_value` (Linking reviews)

Technology Stack & Physical Design
(Combines information from Feb 17 and supplementary, consistent details from Nov 25 doc)

- **Backend:** Python using the Django framework.
- **Frontend:** HTML, Tailwind CSS, JavaScript.
- **Database:** MySQL (RDBMS). Confirmed in both documents.
- **Hosting/Server:** Self-hosted web application on a local server. Uses Django Channels for handling HTTPs and WebSocket communication.
- **Communication:** Standard HTTPs for most requests (CRUD operations). WebSockets for real-time communication (e.g., status updates).
- **Storage:**
    - Primary: Local MySQL database for all application data.
    - Backup: Google Cloud Storage (or similar cloud service) for automated daily backups of critical data.