LeaseLink

Sprint 1 Deliverables

Competitions for this Sprint

Home experience (index.html) with hero messaging and centered marketing copy.

Property catalog (properties.html) pulling real data via backend/get_properties.php.

Rich property detail view (property-details.html) with dynamic backend integration.

Authentication screens (login.html) including role-aware registration workflow.

Forgot password placeholder page (forgot-password.html) wired with interim messaging.

Production-ready database schema (database/leaselink.sql) seeded with sample data.

Work Planned

Persisting session state after login and wiring the UI to backend login.php.

Landlord dashboard for managing listings and booking requests.

Tenant dashboard with saved searches, viewing requests, and messaging.

Search and filter controls on the property catalog page.

Booking workflow and availability management for each property.

Architecture - Major Pages

Home (index.html): Landing page introducing LeaseLink and driving users to browse listings. Status: implemented.

Property Listings (properties.html): Grid of available rentals populated from the properties table. Status: implemented.

Property Details (property-details.html): Deep dive view with dynamic content from get_property.php. Status: implemented, booking CTA still placeholder.

Authentication (login.html): Combined login and registration experience with role selection. Status: implemented UI, backend wiring in progress.

Forgot Password (forgot-password.html): Bootstrap-styled reset flow. Status: implemented placeholder awaiting email backend.

User Dashboards: Role-based landlord and tenant consoles for managing activity. Status: planned for future sprints.

Search and Filters: Catalog refinement controls. Status: planned; depends on query parameter support.

Booking Requests: Viewing scheduling workflow with confirmation states. Status: planned pending API and data model updates.

Backend and API Layer

Backend endpoints are implemented in PHP with mysqli prepared statements to safeguard against SQL injection. The following scripts compose the current API surface:

backend/connect.php - Centralized database connection helper using environment configuration.

backend/get_properties.php - Returns JSON array of listings ordered by created_at.

backend/get_property.php - Fetches a single property by ID with graceful error handling.

backend/login.php - Authenticates landlords or tenants using password_verify.

backend/register.php - Role-aware registration with password strength enforcement and duplicate checks. Planned service additions include password reset endpoints, CRUD operations for properties, booking workflow APIs, and consolidated session management.

Data Model

Current relational schema deployed via database/leaselink.sql:

landlords - Stores landlord profiles with hashed credentials and creation timestamps.

clients - Mirrors landlord structure for renters/tenants.

properties - Core inventory table with descriptive metadata and images.

Envisioned tables to support upcoming features include bookings (property viewing requests), property_images (gallery support), and messages (tenant-landlord communication).

Frontend Stack and Design Choices

The frontend favors lightweight, dependency-free implementation to accelerate early sprint velocity:

- Semantic HTML pages with reusable header/footer structure across views.
- Custom CSS modules (css/global.css, css/properties.css, css/login.css, css/utils.css) leveraging CSS Grid and Flexbox for layout.
- Vanilla JavaScript fetch calls to interact with backend PHP endpoints.
- Bootstrap 5 (CDN) selectively applied on forgot-password.html for rapid form styling.
- Centered typography and call-to-action alignment consistent with prior UX preferences.

No JavaScript frameworks are introduced yet; adopting React or similar is deferred until the dashboard work justifies the overhead.

Risks and Next Steps

- Need session handling and protected routes before dashboards ship.
- Password reset flow requires email infrastructure and security hardening.
- Property images currently rely on external placeholders; need upload/storage strategy.
- Search/filter UX must be designed to avoid performance bottlenecks on large datasets.

Next sprint will prioritize authentication integration on the frontend, landlord CRUD workflows, and early iterations of the dashboard experience.