Software Requirements Specification (SRS)

Project: আমার ঘর – Property Management System

---

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) defines the requirements for আমার ঘর, a bilingual (Bangla/English) property management web application for Bangladesh. It specifies functionality, interfaces, constraints, and quality attributes for developers, testers, and stakeholders.

1.2 Document Conventions

- IEEE-style SRS organization.
- Hierarchical numbering (e.g., 1, 1.1, 1.2).
- Functional requirements are itemized under each feature.

1.3 Intended Audience and Reading Suggestions

- **Developers:** Sections 2, 3, 4, 5.
- **Testers:** Section 3 (System Features).
- **Project Managers/Stakeholders:** Sections 1, 2, 5.
- **Operations/Maintenance Team Leads:** Sections 2.3, 3.5, 4, 5.

1.4 Project Scope

আমার ঘর is a React + Vite web app with Supabase backend. It enables:

- **Tenants**: rent tracking, maintenance requests, messaging.
- **Landlords**: property portfolio, tenant management, financial analytics, reporting.
- **Maintenance Team**: work orders, scheduling, status updates, field notes, and communication.
  The system provides bilingual UI, secure authentication/authorization, and responsive design.

1.5 References

- Supabase Documentation (Auth, PostgreSQL, Storage).
- React & Vite Documentation.
- TailwindCSS Documentation.
- Provided SRS template.

---

2. Overall Description

2.1 Product Perspective

Standalone web application using Supabase for auth, database, and storage. Future integrations may include payments and third-party messaging/notifications.

2.2 Product Features

- **Tenant Dashboard**: rent history, dues, maintenance, messaging.
- **Landlord Dashboard**: properties, tenants, finances, reporting.
- **Maintenance Management**: work order assignment, calendar/scheduling, status tracking, notes/photos.

- **Messaging/Notifications**: secure in-app messages and alerts.
- **Reports & Analytics**: rent performance, occupancy, maintenance KPIs.
- **Bilingual UI** and **Responsive Design**.

2.3 User Classes and Characteristics

- **Tenants**: submit maintenance requests, view rent status, message landlord/maintenance; mobile-first usage.
- **Landlords**: manage properties/tenants, review financials, assign work orders, approve costs; desktop and mobile.
- **Maintenance Team** (Technicians, Supervisors): receive and update work orders, upload photos/notes, reschedule, mark completion; predominantly mobile use, possibly low-connectivity environments.

2.4 Operating Environment

- **Frontend**: Modern browsers (Chrome, Firefox, Safari, Edge).
- **Backend**: Supabase (PostgreSQL, Auth, Storage).
- **Devices**: Desktop, tablet, smartphone.

2.5 Design and Implementation Constraints

- Must support Bangla and English.
- Must use Supabase services.
- Role-based access control (Tenant/Landlord/Maintenance/Admin).
- Responsive design; usable on low-to-mid devices.
- Consider intermittent connectivity for the Maintenance Team.

2.6 User Documentation

- Online user guide and quick start.
- In-app tooltips/tutorials.
- Role-specific FAQs (Tenant, Landlord, Maintenance).

2.7 Assumptions and Dependencies

- Users have stable internet (Maintenance may have intermittent).
- Supabase availability and service limits apply.
- Optional future payment gateway.
- Optional push/SMS/email notifications provider.

---

3. System Features

3.1 Tenant Dashboard

- **Description and Priority**: High. Personalized dashboard for dues, history, maintenance, and communication.
- **Stimulus/Response Sequences**: Tenant logs in → sees rent status; submits request → receives confirmation and updates.
- **Functional Requirements**:
    - Allow tenant login and role-based access.
    - Display rent history, current dues, and payment status.
    - Create, view, and track maintenance requests (title, description, photos, priority).
    - Receive notifications on status changes and messages.
    - View property info and lease terms (read-only).

3.2 Landlord Dashboard

- **Description and Priority**: High. Manage properties, tenants, finances, and maintenance assignments.
- **Stimulus/Response Sequences**: Landlord logs in → sees KPIs; assigns work order → Maintenance Team notified.
- **Functional Requirements**:
    - Create/read/update properties and tenant profiles.
    - View rent collection status and arrears.
    - Generate/export financial and occupancy reports (CSV/PDF).
    - Create/approve/assign maintenance work orders; set priority/SLA/estimated cost.
    - Message tenants and maintenance; receive alerts.

3.3 Messaging and Notifications

- **Description and Priority**: Medium. Secure communication across roles with alerts.
- **Stimulus/Response Sequences**: Message sent → recipient notified → threaded conversation persists.
- **Functional Requirements**:
    - Send/receive messages between Tenant–Landlord–Maintenance.
    - Threaded conversation history with timestamps and read state.
    - Configurable notifications (in-app; optional email/SMS/push in future).
    - Basic attachment support (images for maintenance context).

3.4 Reports & Analytics

- **Description and Priority**: Medium. Financials, occupancy, aging, and maintenance KPIs.

- **Stimulus/Response Sequences**: Landlord requests report → system computes and renders/export.
- **Functional Requirements**:
    - Export to CSV/PDF.
    - Metrics: rent collected vs due, occupancy rate, maintenance turnaround time, SLA compliance, request volume by category.
    - Date range filters and property/portfolio filters.

3.5 Maintenance Management (Maintenance Team Workspace)

- **Description and Priority**: High. End-to-end work order lifecycle for Maintenance Team.
- **Stimulus/Response Sequences**: Work order created/assigned → technician accepts → updates status → adds notes/photos → completes → landlord/tenant notified.
- **Functional Requirements**:
    - Receive and accept/decline assignments; reassign (supervisor).
    - View work order details (description, photos, location, priority, SLA, contact).
    - Update status: **Open** → **Assigned** → **In-Progress** → **On-Hold** → **Completed** → **Verified**.
    - Add field notes, labor time, parts/costs, and before/after photos.
    - Set/adjust schedule; calendar view; detect conflicts.
    - Offline-friendly draft updates (queued submission when back online).
    - Tenant completion confirmation option (optional) and landlord verification.
    - SLA timer tracking and overdue flags.

---

4. External Interface Requirements

4.1 User Interfaces

- Responsive React + TailwindCSS UI.
- Role-based navigation (Tenant/Landlord/Maintenance).
- Language toggle (Bangla/English).
- Accessibility: clear contrast, keyboard navigation, readable fonts.

4.2 Hardware Interfaces

- None beyond standard client devices (desktop/laptop/tablet/phone) and camera usage for maintenance photos.

4.3 Software Interfaces

- Supabase Auth (RBAC), PostgreSQL (data), Storage (images/attachments).
- Optional future: Payment gateway; push/email/SMS provider; analytics SDK.

4.4 Communications Interfaces

- HTTPS for all traffic.
- WebSockets (or Supabase real-time) for live updates/messages.
- Optional webhook endpoints for integrations (future).

---

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- ≤ 2s P95 page response under normal load.
- Support ≥ 1,000 concurrent users initially; scalable to more.
- Real-time updates (messages/status) within 2–5 seconds.

5.2 Safety Requirements

- Regular automated backups of database and storage.
- Graceful error handling and retry for failed operations (especially maintenance uploads).
- Audit logs for critical actions (assignments, status changes).

5.3 Security Requirements

- Role-based access control (Tenant, Landlord, Maintenance, Admin).
- Password hashing; secure session management.
- Data encryption in transit (TLS) and at rest (per backend).
- Access controls on attachments (only authorized roles can view).
- Least-privilege for maintenance users (only assigned work orders).

5.4 Software Quality Attributes

- **Maintainability**: Modular components, clear layering.
- **Scalability**: Horizontal scaling for DB and real-time services.
- **Usability**: Mobile-first flows for Maintenance Team; simple Tenant actions.
- **Reliability**: Error retries, idempotent updates, monitoring hooks.

- **Portability**: Cross-browser/device support.
- **Internationalization**: Full Bangla/English coverage, including dates/numbers.

---

6. Other Requirements

- Compliance with applicable Bangladeshi data protection and privacy norms.
- Data retention policies for messages/work orders (configurable).
- Content moderation policy for attachments/messages (admin tools—future).

Appendices

A. Glossary

- **Work Order**: A maintenance task with details, assignee, schedule, and SLA.
- **SLA**: Target time window for completing a work order.
- **RBAC**: Role-Based Access Control.
- **Maintenance Team**: Technicians and supervisors handling requests.

B. Analysis Models

(Not included in this version; may be provided in design documents.)

C. Issues List

- Choose notification provider(s) (email/SMS/push) for production.
- Offline mode scope for Maintenance app (read vs write caching).
- Payment integration roadmap and fee handling.