cat > /mnt/user-data/outputs/HarmoNet\_機能要件定義書\_v1\_0.md << ‘ENDOFDOC’ # HarmoNet 機能要件定義書

**文書バージョン**: 1.0  
**作成日**: 2025年10月26日  
**最終更新日**: 2025年10月26日  
**プロジェクト名**: HarmoNet（ハーモネット）

## 目次

1. [ドキュメント概要](#X009f0e40f53d6c4cf1ae63441017471e8c887e3)
2. [システム概要](#X7dbc34004d59e548bfc6b15b62a17ea445a49d6)
3. [機能要件](#X751ad3de7fc951ce74e45783dc23d1e982ee8e2)
4. [非機能要件](#X83a17418bdf4339570a6c0197adbbb3ceb16bab)
5. [構成管理・環境要件](#X924b18790543fa8cfdc611a569dd999af253c56)
6. [制約条件・将来展望](#X1ddf01bdd5f3b843922e05547d2d765160adc70)
7. [付録／参照資料](#X1fc1f9896a9abd03e4f683ba023f1c34e2e5c99)

## 1. ドキュメント概要

### 1.1 目的

本書は、HarmoNetアプリケーションの開発における機能および非機能要件を定義し、システム開発・運用に関する共通理解を確立することを目的とする。

### 1.2 適用範囲

本書は、HarmoNetアプリケーションの全体構成および主要機能の要件定義を対象とし、関連するバックエンド・インフラ・UI層を含む。

### 1.3 背景と開発体制

HarmoNetは、地域住民・管理組合・住宅デベロッパー間の情報共有と共助を支援するために構築される。

**開発体制**: MVP方式を採用し、AI協調開発により3.2人月での実装を目指す。

## 2. システム概要

### 2.1 開発目的

HarmoNetは以下の目的を達成するために開発される：

1. **情報伝達の確実化**: 管理組合からのお知らせ・回覧板のデジタル化、多言語対応
2. **共用施設の効率利用**: 駐車場等のオンライン予約システム
3. **コミュニティ活性化**: 住民間の双方向コミュニケーション促進
4. **管理業務の効率化**: 情報発信の一元化、既読状況の可視化
5. **国際化対応**: 日本語・英語・中国語の3言語対応

### 2.2 利用対象者

**一般住民（メインユーザー）**: - 日本人住民（30代〜70代） - 外国人住民（英語圏・中国語圏）

**管理者**: - 管理組合理事 - 管理会社担当者

### 2.3 システム全体像

**技術スタック**: - バックエンド: NestJS (Node.js) - データベース: PostgreSQL - フロントエンド: React / React Native (PWA方式検討中) - スタイリング: Tailwind CSS - コンテナ: Docker - 翻訳: Google Cloud Translation API - 通知: Firebase Cloud Messaging、SendGrid/Amazon SES

**アーキテクチャ**:

[クライアント層]  
 ↓ HTTPS/REST API  
[アプリケーション層 - NestJS]  
 ↓  
[データ層 - PostgreSQL + Redis]  
 ↓  
[外部サービス - Google Translation API等]

### 2.4 運用前提

* **対応デバイス**: スマートフォン（iOS/Android）、タブレット、PC
* **対応ブラウザ**: Chrome、Safari、Firefox、Edge（最新版）
* **運用時間**: 24時間365日（メンテナンス時を除く）
* **稼働率目標**: 99%以上

## 3. 機能要件

### 3.1 ホーム画面

**対象**: MVP  
**概要**: ログイン後に表示されるポータル画面

#### 3.1.1 基本レイアウト

ホーム画面は以下の要素で構成される：

1. **ヘッダーエリア**
   * アプリ名（HarmoNet）
   * 通知アイコン（未読件数バッジ）
   * 言語切替ボタン（JA/EN/CN）
2. **ウェルカムエリア**
   * ログイン中のユーザー名
   * 所属自治体名（例：「A棟-105号室」）
3. **最新お知らせエリア**
   * 最新3件のお知らせ表示
   * 未読/既読状態の表示
   * 「一覧を見る」ボタン
4. **機能タイルエリア**
   * 主要機能へのショートカット（3×3グリッド）
5. **フッターエリア**
   * 固定ナビゲーション

#### 3.1.2 機能タイル一覧

| タイル名 | アイコン | 遷移先 | MVP対象 |
| --- | --- | --- | --- |
| 駐車場予約 | 📅 | 駐車場予約TOP | ○ |
| 回覧板 | 📋 | 回覧板一覧 | ○ |
| 掲示板 | 💬 | 掲示板一覧 | ○ |
| 監視カメラ | 📷 | 監視カメラ参照 | × |
| 見守り | ❤️ | 安否確認 | × |
| 消耗品 | 🛒 | 住居メンテナンス | × |
| メンテナンス | 🔧 | 設備故障報告 | × |
| 通知設定 | 🔔 | 通知設定 | ○ |

### 3.2 掲示板機能

本機能は、住民間の情報共有および管理組合からの周知事項の配信を行うコミュニケーション基盤である。 大規模分譲住宅地における多様な世帯（日本語話者・外国語話者・高齢者・子育て世帯など）を対象とし、心理的な安心感と運用コストの低さを両立させることを目的とする。 特に「荒れないこと」「管理者が疲弊しないこと」を前提条件として設計する。

#### 3.2.1 掲示板の種類

* **全体掲示板（一般掲示板）**  
  全住民が閲覧・投稿可能な掲示板。近隣の気づき、相談、共有事項など日常的な話題を扱う。
* **グループ掲示板**  
  住民が所属するグループ（班・棟・エリアなど）ごとに専用に用意される掲示板。清掃当番・ごみ置き場管理・除草作業など、ローカルな運用連絡を扱う。他グループからは閲覧・投稿不可とし、グループ外への情報拡散を防ぐ。
* **回覧板／重要なお知らせ用掲示枠**  
  管理組合・理事会・管理会社など、公式な立場からのお知らせを掲載する専用枠。掲示は一方向（閲覧のみ）とし、住民からの返信は不可とする。紙の回覧板のデジタル版に相当する。

#### 3.2.2 投稿機能

* **新規スレッド投稿**
  + タイトル、本文、カテゴリ、画像添付に対応。
  + 投稿者は「ニックネーム」または「匿名」を選択可能。匿名でも内部的には投稿者IDを保持し、トレーサビリティを確保。
* **コメント（返信）投稿**
  + 返信はスレッド単位で一段階の階層に限定。複雑な入れ子構造は採用しない。

#### 3.2.3 返信制御ルール

* 投稿には「返信可／返信不可」の属性を設定。
* 一般住民投稿は返信可、管理組合・理事会投稿（「回覧板」タグ）は返信不可。
* 返信可否はタグや投稿種別により自動制御される。

#### 3.2.4 AIモデレーション機能

* 投稿確定前にAIが本文を解析し、不適切表現・攻撃的な文言・個人情報を検知。
* 検知時は注意喚起を表示し、投稿者が修正しない場合でもAIが穏やかな表現に自動変換して投稿。
* 表現の柔和化を目的とし、投稿拒否や検閲は行わない。
* 管理者の負担を軽減しつつ、安心できる対話環境を実現する。

#### 3.2.5 閲覧機能

* カテゴリ表示、タグ表示、コメント件数、翻訳ボタンを備えた一覧画面。
* 詳細画面では本文と画像、コメントを時系列表示。
* 翻訳ボタンで多言語表示（日本語／英語／中国語）に切り替え可能。

#### 3.2.6 編集・削除・通報

* 投稿者本人は自身の投稿を削除可能。
* 管理者はすべての投稿・コメントを削除可能。
* 通報ボタンで不適切投稿を報告でき、理由を選択して送信。
* 管理者による削除ログを保持し、対応の透明性を確保。

#### 3.2.7 回覧板・重要お知らせ機能

* 管理組合・理事会からの公式情報を「回覧板」タグで投稿。
* 返信不可、上部固定表示、「重要」ラベル付与。
* 掲示はお知らせ機能と連携し、全住民へ確実に通知。

#### 3.2.8 既読管理機能（回覧板向け）

* 管理者は回覧板投稿の既読／未読状況を確認可能。
* 各ユーザーごとの閲覧日時を自動記録し、既読率を集計。
* 未読者へ再通知（メール・プッシュ通知）を送信可能。
* CSV出力に対応し、総会報告や運用記録として利用可能。

#### 3.2.9 安心・安全性の担保

* 匿名投稿とAIモデレーションにより、トラブルを未然に防止。
* 投稿者を守りつつ、過剰な言論抑制を避ける設計。
* 管理者の介入を最小限にし、健全な自治と持続的な運用を支援。 ### 3.3 お知らせ表示

**対象**: MVP  
**概要**: 管理組合から住民への一方向情報伝達（Push型）

#### 3.3.1 お知らせ一覧表示（住民側）

**表示項目**: - カテゴリバッジ（重要/イベント/メンテナンス/回覧/その他） - タイトル（ユーザー言語で表示） - 投稿日時 - 未読/既読アイコン

**フィルタ・検索機能**: - カテゴリフィルタ - 未読/既読フィルタ - 日付範囲フィルタ - キーワード検索

#### 3.3.2 お知らせ詳細表示（住民側）

**表示項目**: - カテゴリバッジ - タイトル（多言語対応） - 本文（多言語対応、画像表示対応） - 添付画像（最大5枚） - 投稿日時 - 公開終了日時

**回覧板の追加機能**: - 「確認済み」ボタン - 既読管理（詳細画面を開いた時点で既読）

#### 3.3.3 お知らせ作成・編集（管理者側）

**入力項目**: - タイトル（日本語、最大100文字） - 本文（日本語、最大5000文字、リッチテキスト対応） - カテゴリ（必須） - 公開日時（即時/予約公開） - 公開終了日時（任意） - 添付画像（最大5枚、各5MB以下） - 通知設定（プッシュ/メール）

**翻訳機能**: 1. 日本語で入力 2. 「翻訳」ボタン押下 3. Google Translation APIで翻訳（日→英、日→中） 4. 翻訳結果を確認・修正可能 5. 3言語版を一括保存

**プレビュー機能**: - 住民側の表示イメージをプレビュー - 各言語での表示確認

**公開処理**: 1. 「公開」ボタン押下 2. データベースに保存 3. 通知送信（プッシュ/メール） 4. 公開完了メッセージ

#### 3.3.4 お知らせ管理（管理者側）

**お知らせ一覧（管理者用）**: - 全てのお知らせを一覧表示 - ステータス表示（公開中/予約公開/公開終了） - 回覧板の確認状況集計

**編集・削除**: - 既存お知らせの編集可能 - 論理削除（非表示化）

**既読・確認状況**: - お知らせごとの既読人数・既読率 - 回覧板の未確認者リスト

### 3.4 施設予約

**対象**: MVP（駐車場のみ）、将来実装（集会所等）  
**概要**: 共用施設のオンライン予約システム

#### 3.4.1 対象施設（MVP）

**ゲスト用駐車場**: - 配置: 表側（F1〜F6）6台、裏側（B1〜B6）6台、合計12台 - 利用料金: 1日100円（税込） - 予約単位: 1日単位 - 予約可能期間: 当日から30日先まで - 決済方法: アプリ内決済なし（後日請求）

#### 3.4.2 予約制限ルール

| 制限項目 | ルール |
| --- | --- |
| 同日複数予約 | 同一ユーザーが同日に複数予約不可 |
| 連続予約 | 最大3日間まで連続予約可能 |
| 予約期限 | 当日から30日先まで |
| キャンセル | 利用開始前まで随時可能 |

#### 3.4.3 予約フロー（住民側）

**STEP 1: カレンダー表示** - 当月および翌月のカレンダー - 日付セルの色分け（白:予約可/グレー:満車/緑:自分の予約/赤:過去日）

**STEP 2: 日付選択** - 希望日を選択 - 予約状況を表示

**STEP 3: 駐車場MAP表示** - 表側・裏側をMAP形式で表示 - 各駐車場所の状態を色分け（青:予約可/グレー:予約済/緑:自分の予約）

**STEP 4: 駐車場所選択** - MAP上で希望駐車場所をタップ - 予約済みはタップ不可

**STEP 5: 車両ナンバー入力（任意）** - 下4桁（数字）を入力可能

**STEP 6: 連続予約の選択（任意）** - 連続日数を選択（最大3日間）

**STEP 7: 予約確認画面** - 予約内容の確認 - 「予約を確定する」ボタン

**STEP 8: 予約確定処理** 1. 排他制御による二重予約チェック 2. データベースへ保存 3. 確認メール送信 4. プッシュ通知送信（設定ON時）

**STEP 9: 予約完了画面** - 予約完了メッセージ - 予約内容表示 - 「ホームに戻る」ボタン

#### 3.4.4 予約管理（住民側）

**予約履歴一覧**: - 未来・過去の予約を一覧表示 - ステータス（予約済み/利用完了/キャンセル済み）

**予約詳細**: - 詳細情報表示 - 「キャンセル」ボタン（利用開始前のみ）

**予約キャンセル**: 1. 「キャンセル」ボタン押下 2. 確認ダイアログ 3. キャンセル処理実行 4. キャンセル確認メール送信

#### 3.4.5 通知メール仕様

**送信タイミング**: - 予約完了時 - 予約キャンセル時

**メール内容**: - 件名: 「[HarmoNet] 駐車場予約完了のお知らせ」 - 本文: 予約日時、駐車場所、利用料金、車両ナンバー、注意事項、問い合わせ先

**多言語対応**: - ユーザーの言語設定（JA/EN/CN）に応じて送信

#### 3.4.6 予約管理（管理者側）

**全予約状況の確認**: - カレンダー形式で表示 - 予約詳細（予約者、車両ナンバー等）

**代理予約**: - 管理者が住民に代わって予約可能

**予約のキャンセル・変更**: - 全ての予約をキャンセル可能 - キャンセル理由の記録

**予約ブロック設定**: - 特定日付・駐車場所を「予約不可」に設定 - ブロック理由を入力

**利用統計レポート（将来実装）**: - 月別稼働率 - 人気駐車場所ランキング - CSVエクスポート

### 3.5 掃除当番

**対象**: 将来実装  
**概要**: 共用部清掃の当番表管理

#### 3.5.1 機能概要

* 当番表の自動生成
* 当番日の通知（前日リマインダー）
* 当番完了の報告機能
* 管理者による当番表の編集・調整

*詳細仕様は、MVP完了後に策定する。*

### 3.6 アンケート

**対象**: 将来実装  
**概要**: 管理組合による住民アンケートの実施

#### 3.6.1 機能概要

* アンケート作成（単一選択/複数選択/自由記述）
* 回答期限設定
* 回答状況の集計（リアルタイム）
* 未回答者への自動リマインダー
* 集計結果のグラフ表示・CSVエクスポート

*詳細仕様は、MVP完了後に策定する。*

### 3.7 マイページ

**対象**: MVP  
**概要**: ユーザー設定・履歴確認・管理者機能へのアクセス

#### 3.7.1 管理者機能

管理者ロールを持つユーザーのみがアクセス可能。

##### 3.7.1.1 通知管理

**お知らせ・回覧板の作成・編集・削除**（3.3.3参照） - お知らせの新規作成 - 既存お知らせの編集 - お知らせの削除 - 多言語翻訳 - プレビュー機能

**既読・確認状況の確認**（3.3.4参照） - お知らせごとの既読率 - 回覧板の確認状況

##### 3.7.1.2 ユーザ登録

**新規ユーザーの登録**: - 氏名（必須） - メールアドレス（必須、重複チェック） - 住戸番号（必須） - 言語設定（必須、デフォルト:日本語） - 管理者権限の付与（チェックボックス）

**登録処理**: 1. 入力内容のバリデーション 2. データベースへ保存 3. 初回ログイン用マジックリンクをメール送信

**既存ユーザーの編集・削除**: - ユーザー一覧から選択 - 編集可能項目: 氏名、住戸番号、言語設定、管理者権限 - 削除は論理削除（無効化）

**一括登録機能（CSV インポート）**: - CSV形式で複数ユーザーを一括登録 - フォーマット: 氏名,メールアドレス,住戸番号,言語設定 - エラーレポート出力

##### 3.7.1.3 フッターテーマ設定

**テーマのカスタマイズ**: - フッターナビゲーションの背景色 - アクティブアイコンの色 - アイコンのスタイル

*詳細仕様は、MVP完了後に策定する。*

##### 3.7.1.4 各種マスタ管理

**翻訳マスタ管理**: - 静的UI要素の翻訳編集 - 翻訳キーごとの日英中テキスト管理 - CSV エクスポート/インポート

**カテゴリマスタ管理**: - お知らせのカテゴリ設定 - 掲示板のカテゴリ設定

**施設マスタ管理（将来実装）**: - 予約可能施設の登録 - 施設ごとの予約ルール設定

#### 3.7.2 施設予約履歴

**自分の予約履歴の確認**（3.4.4参照）: - 未来の予約 - 過去の予約 - 予約詳細の確認 - 予約のキャンセル

#### 3.7.3 掲示板投稿履歴

**自分の投稿・コメントの確認**: - 作成したスレッド一覧 - 投稿したコメント一覧 - 投稿の編集・削除

#### 3.7.4 当番表

**自分の当番スケジュールの確認**（将来実装）: - 今後の当番日を一覧表示 - 当番完了報告 - 当番日の変更リクエスト

#### 3.7.5 ユーザ設定

**プロフィール設定**: - 氏名の変更（管理者承認必要） - メールアドレスの変更（確認メール送信） - 言語設定の変更（JA/EN/CN）

**通知設定**: - プッシュ通知のON/OFF（全体、機能別） - メール通知のON/OFF（全体、機能別）

**表示設定**: - 掲示板での表示名: 実名 or 匿名 - ダークモード ON/OFF（将来実装）

**ログアウト**: - ログアウトボタン - 確認ダイアログ - セッション無効化 - ログイン画面へリダイレクト

### 3.8 共通機能

すべての画面で共通して使用される機能。

#### 3.8.1 ヘッダー表示

**構成要素**:

1. **アプリ名（ロゴ）**
   * 左端に「HarmoNet」表示
   * タップでホーム画面へ遷移
2. **通知アイコン**
   * ベルアイコンを右上に表示
   * 未読通知がある場合、バッジに件数表示
   * タップで通知一覧画面へ遷移
3. **言語切替ボタン**
   * 右上に言語ボタン（JA/EN/CN）
   * タップで言語選択ダイアログ
   * 選択した言語で画面を再描画

##### 3.8.1.1 言語切替ボタン

**表示形式**: - ボタンラベル: 「JA」「EN」「CN」 - 選択中の言語は青色でハイライト

**動作**: 1. 言語ボタンをタップ 2. 言語選択ダイアログ表示 3. 言語を選択 4. データベースに保存 5. 画面全体を再描画

**多言語対応の仕組み**: - 静的コンテンツ: 翻訳マスタから取得 - 動的コンテンツ: データベースから取得

#### 3.8.2 コンテンツ表示

**レイアウト**: - レスポンシブデザイン採用 - 画面幅に応じた最適レイアウト - スマートフォン（〜767px）: 1カラム - タブレット（768px〜1023px）: 2カラム - PC（1024px〜）: 最大幅1024px、中央配置

**共通スタイル**: - フォント: システムフォント - 基本カラー: - プライマリ: 青系（#3B82F6） - セカンダリ: 緑系（#10B981） - エラー: 赤系（#EF4444） - 背景: 白/グレー（#F9FAFB） - 角丸: 8px

#### 3.8.3 フッター表示

**構成要素**: - 固定フッターナビゲーション - 5つの主要機能へのショートカット - 現在画面に対応するアイコンをハイライト

##### 3.8.3.1 ショートカットアイコン

**アイコン一覧（MVP）**:

| アイコン | ラベル | 遷移先 |
| --- | --- | --- |
| 🏠 | ホーム | ホーム画面 |
| 🔔 | お知らせ | お知らせ一覧 |
| 💬 | 掲示板 | 掲示板一覧 |
| 📅 | 予約 | 駐車場予約TOP |
| 👤 | マイページ | マイページ |

**ログアウトボタン**: - フッター右端に配置 - アイコン: 🚪 - ラベル: 「ログアウト」（多言語対応） - タップで確認ダイアログ - 「確定」でログアウト処理

**アクティブ状態の表示**: - 現在画面のアイコンを青色に変更 - アイコン下部にインジケーターバー表示

#### 3.8.4 多言語対応

**対応言語**:

| 言語 | 言語コード | 文字種 | 対象ユーザー |
| --- | --- | --- | --- |
| 日本語 | ja / JP | - | 日本人住民 |
| 英語 | en / EN | - | 英語圏住民 |
| 中国語（簡体字） | zh-CN / CN | 簡体字 | 中国本土出身住民 |

*注記: 繁体字対応は将来検討*

**翻訳方式の2層構造**:

**第1層: 静的コンテンツの翻訳** - 対象: UI要素（ボタン、ラベル、メッセージ等） - 実装方式: 翻訳マスタテーブル - テーブル構造:

| カラム | データ型 | 説明 |
| --- | --- | --- |
| translation\_key | VARCHAR | 翻訳キー |
| ja\_text | TEXT | 日本語 |
| en\_text | TEXT | 英語 |
| cn\_text | TEXT | 中国語 |
| category | VARCHAR | カテゴリ |
| updated\_at | TIMESTAMP | 更新日時 |

**第2層: 動的コンテンツの翻訳** - 対象: お知らせ、掲示板投稿等 - 実装方式: Google Cloud Translation API (Basic v2) - API選定理由: 1. サービスの持続性（13年以上の実績） 2. コスト優位性（月50万文字まで無料） 3. 技術的優位性（シンプルなAPI、高速レスポンス）

**翻訳フロー**:

*管理者投稿時*: 1. 日本語で入力 2. 「翻訳」ボタン押下 3. Google Translation APIで翻訳（日→英、日→中） 4. 翻訳結果を確認・修正 5. 3言語版を保存

*住民側の掲示板投稿翻訳*: 1. 異言語の投稿を閲覧 2. 「翻訳」ボタン押下 3. 翻訳リクエスト 4. 翻訳結果をインライン表示 5. 翻訳結果をキャッシュ

**翻訳の最適化**: - 翻訳キャッシュ（再翻訳防止） - 差分翻訳（変更部分のみ） - バッチ処理（まとめて翻訳） - 使用量モニタリング（無料枠の80%でアラート）

**多言語対応の範囲**:

| 対象 | 翻訳方式 | 説明 |
| --- | --- | --- |
| ボタン・ラベル・メニュー | 翻訳マスタ | 事前定義 |
| エラーメッセージ | 翻訳マスタ | 事前定義 |
| システム通知 | 翻訳マスタ | 定型文 |
| お知らせ・イベント情報 | Google Translation API | 動的翻訳 |
| 利用規約・ヘルプ | Google Translation API | 初回翻訳、以降固定 |
| 掲示板投稿 | Google Translation API | ボタン押下時 |
| メール通知本文 | 翻訳マスタ + 動的挿入 | テンプレート+変数 |

**日付・数値表記の国際化**:

| 項目 | 日本語 | 英語 | 中国語 |
| --- | --- | --- | --- |
| 日付形式 | 2025年10月26日 | October 26, 2025 | 2025年10月26日 |
| 曜日 | 月火水木金土日 | Mon-Sun | 周一-周日 |
| 通貨表示 | ¥100 | ¥100 (JPY) | ¥100 |

**コスト試算**:

| 項目 | 文字数 | 頻度 | 月間合計 |
| --- | --- | --- | --- |
| お知らせ投稿 | 500文字 | 月10件 | 5,000文字 |
| イベント情報 | 300文字 | 月5件 | 1,500文字 |
| 予約通知メール | 200文字 | 月100件 | 20,000文字 |
| 掲示板投稿翻訳 | 100文字 | 月100回 | 10,000文字 |
| **合計** |  |  | **36,500文字/月** |

**結論**: Google Translation APIの無料枠（月50万文字）内で運用可能。コストは実質¥0。

### 3.9 通知機能

**対象**: MVP  
**概要**: 住民への情報伝達手段

#### 3.9.1 アプリ通知（プッシュ通知）

**対象イベント**: - お知らせ・回覧板の新規公開 - 駐車場予約の完了・キャンセル - 掲示板の自分のスレッドへのコメント - リマインダー（予約前日等）

**通知内容**: - タイトル: イベント応じた短いメッセージ - 本文: 詳細情報 - アクション: タップで該当画面へ遷移

**多言語対応**: - ユーザーの言語設定に応じた言語で通知

**配信制御**: - ユーザーの通知設定に従う - 機能別ON/OFF設定対応

**技術実装**: - Firebase Cloud Messaging (FCM) - iOS・Android両対応

#### 3.9.2 メール通知

**対象イベント**: - マジックリンク送信（認証用） - お知らせ・回覧板の新規公開 - 駐車場予約の完了・キャンセル - 掲示板の自分のスレッドへのコメント - リマインダー

**メール内容**: - 差出人: HarmoNet [noreply@harmonet.example.com](mailto:noreply@harmonet.example.com) - 件名: イベント応じた件名 - 本文: - テキスト形式 - 詳細情報 - アプリへのリンク - フッター: 配信停止リンク、問い合わせ先

**多言語対応**: - ユーザーの言語設定に応じた言語でメール送信 - メールテンプレートは翻訳マスタで管理

**配信制御**: - ユーザーの通知設定に従う

**技術実装**: - SendGridまたはAmazon SES - SPF/DKIM/DMARC設定

## 4. 非機能要件

### 4.1 セキュリティ

#### 4.1.1 認証方式

**マジックリンク方式（パスワードレス認証）**

HarmoNetでは、パスワードを使用しないマジックリンク方式を採用。

**メリット**: - パスワード管理の負担軽減 - パスワード漏洩リスクの排除 - ユーザビリティの向上

**認証フロー**:

1. **ログイン要求**
   * ユーザーがメールアドレスを入力
   * 「ログインリンクを送信」ボタン押下
2. **メールアドレス確認**
   * ユーザーマスタで存在確認
   * 存在しない場合でも「メールを送信しました」と表示（アドレス列挙攻撃対策）
3. **マジックリンク生成**
   * 予測困難なランダムトークン（UUID v4）を生成
   * トークンをデータベースに保存（有効期限:15分、使用回数:1回）
   * 認証用URLを生成
4. **メール送信**
   * 認証用URLを含むメールを送信
   * 注意書き: 「心当たりのない場合は無視してください」
5. **認証実行**
   * ユーザーがリンクをクリック
   * トークンの有効性を検証（有効期限内、未使用、存在）
   * 有効な場合、セッショントークンを発行
   * トークンを「使用済み」に変更
6. **ホーム画面へ遷移**

**セキュリティ考慮事項**: 1. レートリミット: 1分間に3回まで 2. トークンの安全性: 128ビット以上のランダム性、HTTPS必須 3. セッション管理: HTTPOnly Cookie、Secure属性、SameSite=Strict、有効期限7日間 4. 監査ログ: ログイン試行をすべて記録（IPアドレス、デバイス情報含む） 5. 強制ログアウト: 管理者が特定ユーザーのセッションを無効化可能

#### 4.1.2 暗号化方式

**通信の暗号化**: - すべての通信はHTTPS（TLS 1.3推奨、最低TLS 1.2） - HTTPアクセスは拒否（HTTPSへリダイレクト）

**データの暗号化**: - 個人情報: AES-256で暗号化保存（氏名、メールアドレス、住戸番号） - セッショントークン: SHA-256でハッシュ化 - APIキー: 環境変数で管理、コード内にハードコーディングしない

**鍵管理**: - 暗号化鍵は環境変数で管理 - 本番環境では秘密管理サービス使用（AWS Secrets Manager等） - 定期的な鍵ローテーション（年1回）

### 4.2 性能要件

| 項目 | 要件 |
| --- | --- |
| ページ読み込み時間 | 3秒以内（初回）、1秒以内（2回目以降） |
| 翻訳処理時間 | 5秒以内（500文字） |
| 同時アクセス対応 | 100ユーザー（MVP）、将来的に500ユーザー |
| DBクエリ応答時間 | 1秒以内（95パーセンタイル） |
| API応答時間 | 2秒以内（95パーセンタイル） |

**パフォーマンス最適化施策**: - 画像の最適化（WebP形式、遅延読み込み） - CDN利用 - データベースインデックス最適化 - Redis活用（翻訳キャッシュ、セッション管理） - コード分割（React Lazy Loading）

### 4.3 アクセシビリティ

**WCAG 2.1 レベルAA準拠**: - 視覚: コントラスト比4.5:1以上、拡大表示対応（200%まで） - 聴覚: 動画コンテンツには字幕（将来実装時） - 操作: キーボードのみで全操作可能、フォーカスインジケーター明示 - 理解: 明確なエラーメッセージ、一貫性のあるナビゲーション

**多言語・国際化**: - 日本語・英語・中国語対応 - RTL言語は非対応（将来検討）

**モバイルアクセシビリティ**: - タップ領域: 最小44×44px - スワイプジェスチャー対応 - 画面回転対応

### 4.4 スケーラビリティ

**ユーザー数の増加対応**: - 初期: 100ユーザー（単一マンション） - 中期: 500ユーザー（複数マンション） - 長期: 5,000ユーザー（複数デベロッパー）

**スケーリング戦略**: - 垂直スケーリング: サーバースペック増強 - 水平スケーリング: - アプリケーションサーバーの複数台構成 - データベースのレプリケーション - キャッシュサーバーの複数台構成 - マイクロサービス化: 将来的に機能ごとにサービス分割

**マルチテナント対応**: - 複数の管理組合を1つのアプリで管理 - データの論理的分離（tenant\_id） - 管理組合ごとの独立したカスタマイズ設定

## 5. 構成管理・環境要件

### 5.1 開発環境（NestJS, PostgreSQL, Tailwind, Docker）

**技術スタック**:

| レイヤー | 技術 | バージョン |
| --- | --- | --- |
| バックエンド | NestJS (Node.js) | 10.x |
| データベース | PostgreSQL | 15.x |
| キャッシュ | Redis | 7.x |
| フロントエンド | React / React Native | 18.x / 0.72.x |
| スタイリング | Tailwind CSS | 3.x |
| コンテナ | Docker | 24.x |
| オーケストレーション | Docker Compose | 2.x |

**開発ツール**: - エディタ: Visual Studio Code - バージョン管理: Git - パッケージマネージャー: npm / yarn - APIテスト: Postman / Insomnia - データベース管理: pgAdmin / A5:SQL Mk-2

**ローカル開発環境構築**: 1. Docker Desktopのインストール 2. リポジトリのクローン 3. 環境変数ファイル（.env）の設定 4. docker-compose up -d でコンテナ起動 5. npm install で依存パッケージインストール 6. npm run migration:run でDB初期化 7. npm run seed でマスタデータ投入 8. npm run dev で開発サーバー起動

**開発規約**: - コーディング規約: Airbnb JavaScript Style Guide - コミットメッセージ: Conventional Commits - ブランチ戦略: Git Flow - コードレビュー: プルリクエストによるピアレビュー必須

### 5.2 運用環境・ホスティング構成

**ホスティング候補**:

1. **AWS (Amazon Web Services)** - 推奨
   * ECS (Elastic Container Service)
   * RDS (Relational Database Service) for PostgreSQL
   * ElastiCache for Redis
   * S3 で画像保存
   * CloudFront (CDN)
2. **GCP (Google Cloud Platform)**
   * Cloud Run
   * Cloud SQL for PostgreSQL
   * Memorystore for Redis
   * Cloud Storage
   * Cloud CDN
3. **Vercel + Supabase** - 低コスト案
   * Vercel でフロントエンド・APIホスト
   * Supabase でPostgreSQL + 認証
   * Cloudflare R2 で静的ファイル保存

**推奨構成（AWS）**:

Internet  
 ↓  
CloudFront (CDN)  
 ↓  
ALB (Application Load Balancer)  
 ↓  
ECS (Fargate) - NestJSアプリ（複数コンテナ）  
 ↓  
RDS (PostgreSQL) - マスター  
 ↓  
RDS (PostgreSQL) - 読み取りレプリカ（将来）  
 ↓  
ElastiCache (Redis)  
 ↓  
S3 - 画像保存

**環境分離**: - 開発環境: ローカルDocker - ステージング環境: 本番同等構成 - 本番環境: AWS/GCP本番環境

### 5.3 バックアップ／監視設計

**バックアップ戦略**:

| 対象 | 頻度 | 保管期間 | 保管場所 |
| --- | --- | --- | --- |
| データベース（完全） | 日次（深夜2:00） | 7日分 | S3 / Cloud Storage |
| データベース（週次） | 週次（日曜深夜） | 4週分 | S3 / Cloud Storage |
| データベース（月次） | 月次（月末深夜） | 12ヶ月分 | S3 Glacier / Archive |
| 画像・ファイル | リアルタイム | 30日分 | S3 Versioning |
| アプリケーションコード | Gitコミットごと | 永続 | GitHub / GitLab |

**リカバリ手順**: 1. バックアップファイルの特定 2. ダウンロードと検証 3. ステージング環境でリストア・動作確認 4. 本番環境でリストア 5. アプリケーション再起動 6. 動作確認

**監視項目**:

| カテゴリ | 監視項目 | アラート条件 |
| --- | --- | --- |
| サーバー | CPU使用率 | 80%以上が5分継続 |
| サーバー | メモリ使用率 | 85%以上が5分継続 |
| サーバー | ディスク使用率 | 90%以上 |
| アプリケーション | エラー発生率 | 5%以上 |
| アプリケーション | 応答時間 | 3秒以上が5分継続 |
| データベース | 接続数 | 最大の80%以上 |
| データベース | スロークエリ | 3秒以上 |
| 外部API | Google Translation API使用量 | 無料枠の80%到達 |

**監視ツール**: - アプリケーション監視: Datadog / New Relic / CloudWatch - エラートラッキング: Sentry - ログ管理: CloudWatch Logs / Stackdriver Logging - アラート通知: Slack / メール

**稼働率目標**: - 稼働率: 99%以上（年間停止時間: 約87時間以内） - 計画メンテナンス: 月1回、深夜（2:00〜4:00）、2時間以内

## 6. 制約条件・将来展望

### 6.1 開発スコープ制限

**MVP（初期リリース）の制約**:

1. **決済機能非対応**
   * 駐車場利用料金はアプリ内決済なし
   * 管理費と合算して後日請求
   * 将来的に決済機能追加を検討
2. **中国語は簡体字のみ対応**
   * 繁体字（台湾、香港）は将来検討
   * 他言語（韓国語、ベトナム語等）も将来検討
3. **時間単位の予約非対応**
   * 駐車場予約は1日単位のみ
   * 集会所等の時間単位予約は将来実装
4. **マルチテナント機能は限定的**
   * 初期リリースは単一物件のみ
   * 複数物件対応は将来実装
5. **Google Translation APIの無料枠依存**
   * 月50万文字まで無料
   * 超過時は従量課金
6. **同時接続ユーザー数制限**
   * 初期構成: 100ユーザー
   * スケーリングで500ユーザーまで対応

### 6.2 将来的な拡張（AI連携・スマート通知）

**フェーズ2（MVP完了後6ヶ月以内）**:

1. **安否確認機能**
   * 災害発生時の一斉安否確認
   * 回答状況の集計・可視化
   * 未回答者への自動リマインダー
2. **監視カメラ参照機能**
   * 共用部の監視カメラ映像のライブ配信
   * 録画映像の参照
3. **住居メンテナンス機能**
   * 住宅設備機器の製品カタログ
   * オンライン注文機能
4. **アンケート機能**
   * 質問形式: 単一選択、複数選択、自由記述
   * 集計結果のグラフ表示
5. **掃除当番機能**
   * 当番表の自動生成
   * 当番日の通知

**フェーズ3（MVP完了後1年以内）**:

1. **HEMS連携**
   * Home Energy Management Systemと連携
   * 電力使用量のグラフ表示
2. **生活サポート機能**
   * ゴミ出しカレンダー
   * 管理会社への連絡（設備故障報告）
   * 宅配ボックス利用通知
3. **管理・経理機能**
   * 管理費等の確認
   * オンライン決済機能
   * 総会資料の閲覧
4. **AI連携機能**
   * AIチャットボット（よくある質問への自動回答）
   * お知らせの自動要約
   * 掲示板投稿の自動翻訳品質向上（GPT-4等活用）
   * 異常検知（長期間ログインなし等）
5. **スマート通知**
   * ユーザーの行動パターンに基づく最適な通知タイミング
   * 通知の優先度自動判定
   * 未読通知の自動リマインダー

**フェーズ4（MVP完了後2年以内）**:

1. **マルチテナント本格対応**
   * 複数物件の一元管理
   * 物件ごとのカスタマイズ設定
   * デベロッパー向け管理ダッシュボード
2. **スマートホーム連携**
   * スマートロック連携
   * 温度・湿度センサー連携
   * IoTデバイス管理
3. **コミュニティ機能の拡充**
   * イベント企画・募集機能
   * 物品の貸し借り機能
   * 住民スキルシェアリング
4. **データ分析・レポート**
   * 各機能の利用統計
   * 住民満足度調査
   * 予測分析（駐車場混雑予測等）

## 7. 付録／参照資料

### 7.1 用語集

| 用語 | 説明 |
| --- | --- |
| HarmoNet | 本アプリケーションの正式名称 |
| MVP | Minimum Viable Product（実用最小限の製品） |
| マジックリンク | パスワードレス認証方式 |
| 翻訳マスタ | UIの固定文言の多言語翻訳を定義したテーブル |
| Google Translation API | Googleが提供する機械翻訳API |
| 全体板 | 全住民が閲覧・投稿可能な掲示板 |
| グループ板 | 所属グループのみが閲覧・投稿可能な掲示板 |
| Push型 | システムからユーザーへ情報を能動的に送信 |
| Pull型 | ユーザーが能動的に情報を取得 |
| レートリミット | 短時間での過度なリクエストを制限 |
| 排他制御 | 複数ユーザーの同時更新を防ぐ仕組み |

### 7.2 関連ドキュメント

| ドキュメント名 | 概要 |
| --- | --- |
| HarmoNet 画面設計書 | 詳細レイアウト、UI要素、画面遷移 |
| HarmoNet データベース設計書 | テーブル定義、ER図、インデックス設計 |
| HarmoNet API仕様書 | RESTful APIのエンドポイント、リクエスト/レスポンス |
| HarmoNet インフラ構成図 | サーバー構成、ネットワーク構成 |
| HarmoNet セキュリティガイドライン | セキュリティポリシー、脆弱性対策 |
| HarmoNet 運用マニュアル | 日常運用、トラブルシューティング |
| HarmoNet ユーザーマニュアル | 住民向け使い方ガイド（多言語） |

### 7.3 外部サービス・API一覧

| サービス名 | 用途 | 料金 |
| --- | --- | --- |
| Google Cloud Translation API (Basic v2) | 動的コンテンツの自動翻訳 | 月50万文字まで無料超過時: $20/100万文字 |
| Firebase Cloud Messaging | プッシュ通知配信 | 基本無料 |
| SendGrid | メール配信 | 月100通まで無料超過時: $19.95/月（40,000通） |
| Amazon SES | メール配信（代替案） | $0.10/1,000通 |

### 7.4 開発スケジュール（暫定）

| フェーズ | 期間 | 主な成果物 |
| --- | --- | --- |
| 要件定義 | 2025年10月 | 本要件定義書の確定 |
| 静的プロトタイプ作成 | 2025年11月 | 全画面の静的HTMLデモ完成 |
| 基本設計 | 2025年12月 | 画面設計書、DB設計書、API設計書 |
| 詳細設計・開発 | 2026年1月〜3月 | MVP機能の実装完了 |
| テスト・調整 | 2026年3月〜4月 | 品質確認、バグ修正 |
| MVP リリース | 2026年5月 | 初期リリース、住民への公開 |
| フィードバック収集・改善 | 2026年5月〜7月 | 利用状況分析、機能改善 |
| 機能拡張（第2フェーズ） | 2026年8月以降 | 将来実装機能の追加開発 |

### 7.5 リスク管理

| リスク | 発生確率 | 影響度 | 対策 |
| --- | --- | --- | --- |
| Google Translation API 使用量超過 | 低 | 中 | 使用量監視、キャッシュ活用 |
| セキュリティインシデント | 中 | 高 | 多層防御、監査ログ、脆弱性診断 |
| 大量アクセスによるサーバーダウン | 低 | 高 | オートスケーリング、CDN、ロードテスト |
| データベース障害 | 低 | 高 | レプリケーション、定期バックアップ |
| 開発スケジュール遅延 | 中 | 中 | MVP優先順位明確化、段階的リリース |
| 住民の利用率が低い | 中 | 高 | ユーザビリティテスト、説明会開催 |

**文書承認**

| 役割 | 氏名 | 承認日 |
| --- | --- | --- |
| プロジェクトオーナー | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **/**/\_\_ |
| 開発リーダー | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **/**/\_\_ |

**変更履歴**

| バージョン | 日付 | 変更内容 | 変更者 |
| --- | --- | --- | --- |
| 1.0 | 2025/10/26 | 初版作成 | TKD |

**重要事項**

本要件定義書の内容は、実装前に関係者全員（管理組合理事会、管理会社、開発チーム）でレビューし、合意を得る必要があります。また、実装過程で新たな要件が発覚した場合は、本文書を更新し、バージョン管理を行います。

本資料は機密情報を含みます。無断での複製・配布を禁じます。

**文書管理番号**: HARMONET-REQ-001  
**文書分類**: 要件定義書  
**セキュリティレベル**: 社外秘 ENDOFDOC

**更新履歴**

* v1.1 (2025-10-26): 第3章 掲示板機能を改訂（AIモデレーション、グループ掲示板、匿名投稿、回覧板・既読管理機能を追加）