

instaVRam プロジェクト仕様書

プロジェクト概要

instaVRam は、Next.js (App Router) + Firebase + Mantine UI を用いたフォト共有 SNS アプリケーションです。ユーザー認証、アルバム作成、画像アップロード、コメント、いいね、リポスト、リアクション、フレンド申請、ウォッチ機能、通知、検索などの機能を備えています。

技術スタック:

- **フロントエンド:** Next.js 16.1 (App Router), React 19.2, TypeScript 5
- **バックエンド/DB:** Firebase Authentication, Firestore, Storage, Admin SDK
- **UI:** Mantine 7, Tailwind CSS 4
- **画像処理:** react-easy-crop (アバター切り抜き), react-image-crop, lightgallery (ギャラリービューアー)
- **画像表示:** react-photo-album (レスポンシブグリッド)
- **テスト:** Jest 29.7, React Testing Library 16.3
- **その他:** Nodemailer 7.0 (メール送信), react-intersection-observer (無限スクロール)

ディレクトリ構成とファイル説明

ルートディレクトリ

```
virtualbum/
├── app/
│   ├── album/[id]/
│   │   ├── hooks/
│   │   └── page.tsx
│   ├── user/[id]/
│   │   ├── hooks/
│   │   └── page.tsx
│   └── api/           # API Routes
├── components/
└── lib/
    ├── errors/        # エラーハンドリング統一モジュール
    ├── repos/         # Firestore リポジトリ層 (Client SDK)
    ├── constants/     # 定数定義
    ├── auth/          # 認証関連
    └── utils/         # ユーティリティ関数
src/
    ├── hooks/         # カスタム React Hooks
    ├── services/      # ビジネスロジックサービス
    ├── repositories/  # Firestore リポジトリ層 (Admin SDK)
    │   └── admin/      # Admin SDK を使用するリポジトリ
    ├── libs/          # サードパーティライブラリラッパー
    ├── models/         # ピューモデル定義
    ├── types/          # アプリケーション層の型定義
    └── components/    # src 内部で使う独立コンポーネント
    types/             # 外部ライブラリの型定義 (.d.ts)
```

```

└── test/          # テストファイル
└── scripts/      # シードスクリプト等
└── design/       # 設計ドキュメント
└── functions/    # Firebase Functions
└── firestore.rules # Firestore セキュリティルール
└── storage.rules  # Storage セキュリティルール
└── package.json   # npm 依存関係
└── tsconfig.json  # TypeScript 設定

```

ルート直下の設定ファイル

Firebase 関連

ファイル	説明
<code>firebase.json</code>	Firebase エミュレータ設定 (Auth:9099, Firestore:8080, Storage:9199, UI:4000)
<code>firestore.rules</code>	Firestore セキュリティルール定義
<code>firestore.indexes.json</code>	Firestore 複合インデックス定義
<code>storage.rules</code>	Cloud Storage セキュリティルール定義
<code>.firebaserc</code>	Firebase プロジェクトエイリアス設定

ビルド・開発ツール

ファイル	説明
<code>next.config.ts</code>	Next.js 設定 (Turbopack、画像ドメイン許可等)
<code>tsconfig.json</code>	TypeScript 設定 (ES2017、strict モード、パスエイリアス <code>@/*</code>)
<code>eslint.config.mjs</code>	ESLint 設定 (next/core-web-vitals ベース)
<code>postcss.config.mjs</code>	PostCSS 設定 (Tailwind CSS 用)

テスト

ファイル	説明
<code>jest.config.js</code>	Jest 設定 (jsdom 環境、パスエイリアス対応)
<code>jest.setup.ts</code>	Jest セットアップ (Testing Library、Next.js モック)

パッケージ管理

ファイル	説明
<code>package.json</code>	npm 依存関係とスクリプト定義

ファイル	説明
package-lock.json	依存関係のロックファイル

環境・Git

ファイル	説明
.env.local	ローカル環境変数（Firebase キー等、Git 管理外）
.gitignore	Git 除外設定（node_modules、.next、.env.local 等）

app/ - Next.js App Router ページ構成

app/layout.tsx

ルートレイアウトファイル。Mantine の ColorSchemeScript、ThemeProvider、SideNav、AuthGate を組み込み、全ページ共通の構造を提供します。

app/page.tsx

トップページ（ログイン前）。AuthForm コンポーネントを表示し、ログイン/新規登録へ誘導します。

app/providers.tsx

Mantine の MantineProvider でテーマとカラースキームを設定。全ページに React Context を提供します。

app/globals.css

Tailwind CSS の基本スタイルとカスタム変数（カラースキーム等）を定義します。

app/timeline/page.tsx (628 lines)

メインタイムライン画面。ログインユーザーの投稿、フレンドの投稿、ウォッチ中のユーザーの投稿を時系列に表示します。無限スクロール対応、リアルタイム購読（コメント、いいね、リポスト）、オプティミステイック更新を実装しています。

app/album/[id]/page.tsx (1526 lines)

アルバム詳細ページ。アルバムタイトル・URL・公開設定の編集、画像一覧、コメント入力、いいね・リポスト・リアクション操作、参加ユーザー表示を行います。lightgallery で画像ギャラリー表示にも対応します。

app/album/new/page.tsx

新規アルバム作成ページ。ImageUploadFlow コンポーネントでタイトル、公開設定を指定して複数画像を一度にアップロードできます。

app/user/[id]/page.tsx (1526 lines)

ユーザープロフィール画面。 プロフィール編集（handle, displayName, bio, icon, cover, banner） 、フレンド申請/承認/解除、ウォッч追加/削除、アルバム/参加アルバム/コメント/画像タブ表示、アカウント削除フォームを提供します。

app/search/page.tsx

検索画面。 ユーザー検索（handle/displayName）に対応し、結果をリスト表示します。将来的にアルバム検索も拡張予定です。

app/notification/page.tsx

通知一覧画面。 コメント、いいね、フレンド申請、ウォッチ、リポスト、リアクションの通知をリアルタイム購読で表示します。既読一括マークにも対応します。

app/forgot-password/page.tsx

パスワード忘れページ。 メールアドレスを入力してパスワードリセットメールを送信します。中立的なメッセージでアカウント存在の列挙攻撃を防止します。

app/reset-password/page.tsx

パスワードリセットページ。 メールから送られたリンクをクリックして新しいパスワードを設定します。

app/login/page.tsx

ログインページ。 メールアドレス+パスワードでログインします。エラーメッセージ表示とログイン後リダイレクトを処理します。

app/register/page.tsx

新規登録ページ。 メールアドレス、パスワード、ハンドル、表示名を入力してアカウントを作成します。handle の重複チェック付き。

app/register/complete/page.tsx

登録完了画面。 登録後、メール確認の案内を表示します。

app/api/ - サーバー API Routes

app/api/comments/add/route.ts

コメント追加 API。クライアント側から fetch して Firestore にコメントを作成します。

app/api/likes/toggle/route.ts

いいねトグル API。Admin SDK でいいね作成/削除を実行します。未初期化時はクライアント SDK ヘフォールバック。

app/api/reposts/toggle/route.ts

リポストトグル API。likes と同様、Admin SDK 優先でリポスト ON/OFF を切り替えます。

`app/api/reactions/toggle/route.ts`

リアクションのトグル API。絵文字単位でリアクション追加/削除を行います。

`app/api/images/add/route.ts`

画像追加 API。アルバムへの画像追加を Admin SDK で実行します。

`app/api/images/delete/route.ts`

画像削除 API。指定 `imageId` の画像を削除し、Storage からも削除を試みます。

`app/api/share/discord/route.ts`

Discord Webhook 経由で投稿共有を行う予定のエンドポイント。

`app/api/reports/album/route.ts`

アルバム通報 API。将来的にモデレーション機能に対応予定。

`components/` - UI コンポーネント

`components/AuthForm.tsx`

ログインフォーム。メールアドレス+パスワードでログインまたは新規登録へのリンクを提供します。

`components/AuthGate.tsx`

認証ガード。未ログイン時にログインページへリダイレクトします。

`components/SideNav.tsx`

サイドナビゲーション。タイムライン、通知、検索、プロフィール、新規アルバム作成へのリンクを表示します。

`components/Header.tsx, components/HeaderGate.tsx`

ヘッダー部品。将来的にトップバー表示に使用予定。

`components/ThemeSwitch.tsx`

ライトモード/ダークモード切り替えスイッチ（将来実装予定）。

`components/AlbumCard.tsx`

アルバムカード UI。画像サムネイル、タイトル、オナー情報を表示します。

components/AlbumCreateModal.tsx

アルバム作成ダイアログ。簡易的なモーダル UI でタイトルと公開設定を入力します。

components/album/ - アルバム関連 UI

components/album/AlbumActionsMenu.tsx

アルバム操作メニュー（編集・削除・通報）。オーナーと閲覧者で表示項目を切り替えます。

components/album/DeleteConfirmModal.tsx

削除確認ダイアログ。アルバムまたは画像削除時に確認を求めます。

components/album/ReportConfirmModal.tsx

通報確認ダイアログ。アルバム通報機能に使用します。

components/album/ShareMenu.tsx

共有メニュー。URL コピー、Twitter 共有、Discord 共有（実装予定）ボタンを提供します。

components/comments/ - コメント UI

components/comments/CommentItem.tsx

コメント 1 件の表示 UI。ユーザーアイコン、本文、削除ボタン（本人またはアルバムオーナーのみ）を表示します。

components/comments/CommentInput.tsx

コメント入力フォーム。textarea でコメントを入力し、送信ボタンで投稿します。

components/form/ - 汎用フォームコンポーネント

components/form/TextareaAutosize.tsx

自動リサイズする textarea コンポーネント（コメント入力等で使用）。

components/gallery/ - ギャラリー関連 UI

components/gallery/ImageGrid.tsx

画像グリッド表示。クリックでギャラリービューアーを開きます。

components/gallery/GalleryViewer.tsx

lightgallery を使ったフルスクリーン画像ビューアー（スワイプ対応）。

components/profile/ - プロフィール UI

components/profile/Avatar.tsx

ユーザーアイコン表示コンポーネント。サイズ指定で動的にスタイルを変更します。

components/profile/AvatarModal.tsx

アバター編集ダイアログ。react-easy-crop で画像をトリミングして Storage ヘアップロードします。

components/profile/AvatarCropper.tsx

トリミング UI 本体。react-easy-crop の操作 UI を提供します。

components/profile/FriendActions.tsx

フレンドアクション UI。申請送信、承認、解除ボタンを状態に応じて表示します。

components/profile/FriendRemoveConfirmModal.tsx

フレンド解除確認ダイアログ。解除時に確認を求めます。

components/profile/WatchActions.tsx

ウォッチ（フォロー）操作ボタン。ウォッチ追加/削除を切り替えます。

components/timeline/ - タイムライン UI

components/timeline/TimelineItem.tsx (250 lines)

タイムライン 1 行のメインコンポーネント。サブコンポーネントを組み合わせて構成します。

components/timeline/types.ts (80 lines)

共通型定義 (Img, UserRef, TimelineItemProps, ReactionData 等)。

components/timeline/utils.ts (40 lines)

ユーティリティ関数 (toDate, formatDate, truncateText)。

components/timeline/ImageGrid.tsx (130 lines)

1~4枚の画像を最適なレイアウトで表示するグリッドコンポーネント。

components/timeline/ActionBar.tsx (200 lines)

いいね/リポスト/コメントボタンのアクションバー。ホバー時にユーザー一覧をポップオーバー表示。

components/timeline/ReactionSection.tsx (335 lines)

リアクションチップと絵文字ピッカー。カテゴリ別絵文字選択、リアクター表示に対応。

components/timeline/CommentPreview.tsx (65 lines)

最新コメント3件のプレビュー表示。

components/timeline/UserListPopover.tsx (85 lines)

いいね/リポスト/リアクションのユーザー一覧ポップオーバー（共通コンポーネント）。

components/timeline/index.ts (10 lines)

サブコンポーネントの統一エクスポート。

components/ui/ - 汎用 UI コンポーネント

components/ui/Button.tsx

汎用ボタンコンポーネント。variant (primary, ghost, danger 等) とサイズを指定可能です。

components/ui/Toast.tsx

トースト通知 UI。成功・エラーメッセージを画面右上に表示します。

components/ui/ThemeSwitch.tsx

テーマ切り替えスイッチ（実装予定）。

components/upload/ - アップロード UI

components/upload/ImageUploadFlow.tsx

画像アップロードフロー。複数画像選択、タイトル・公開設定入力、一括アップロード処理を統合します。

components/upload/ImagePreviewGrid.tsx

アップロード前の画像プレビューグリッド表示。

components/upload/ImageDropzone.tsx

ドラッグ＆ドロップ対応の画像選択 UI。Mantine Dropzone を使用します。

components/icons/ - アイコンコンポーネント

components/icons/HeartIcon.tsx

いいねアイコン（ハートマーク）。塗りつぶしでアクティブ状態を表現します。

components/icons/ChatIcon.tsx

コメントアイコン（吹き出しマーク）。

`components/icons/RepostIcon.tsx`

リポストアイコン（リツイートマーク）。

`components/icons/SearchIcon.tsx`

検索アイコン。

`components/icons/BellIcon.tsx`

通知アイコン（ベルマーク）。

`components/icons/UserIcon.tsx`

ユーザーアイコン（人マーク）。

`lib/` - ビジネスロジック・リポジトリ層

`lib/firebase.ts`

Firebase アプリ初期化と db, auth, storage のエクスポート。環境変数から設定を読み込みます。

`lib/logger.ts`

環境に応じたロガーユーティリティ。production では error のみ、development では全て出力。`createLogger(prefix)` で名前付きロガーを作成できます。

`lib/paths.ts`

Firebase コレクション名を定数として集中管理（`COL.users`, `COL.albums` など）。

`lib/rateLimit.ts`

レート制限関連のユーティリティ。`isRateLimitError` でレート制限エラーを判定します。

`lib/authUser.ts`

認証ユーザー関連のユーティリティ。

`lib/errors/` - エラーハンドリング統一モジュール

`lib/errors/index.ts`

エラーハンドリングの統一エクスポート。`translateError`（シンプルなエラー翻訳）、`AppError`、`translateFirebaseError`、`handleError`、`ErrorHelpers` を提供。

`lib/errors/ErrorHandler.ts`

構造化されたエラーハンドリング。カスタムエラークラス `AppError` と `Toast` 連携。

`lib/constants/` - 定数定義

`lib/constants/reactions.ts`

リアクション絵文字のカテゴリ定義（顔、ハート、食べ物、スポーツ等）。

`lib/repos/` - Firestore アクセス層

`lib/repos/userRepo.ts`

ユーザー情報の CRUD 操作（作成、取得、更新、検索）。ハンドル重複チェック機能付き。

`lib/repos/albumRepo.ts`

アルバムの作成、取得、更新、削除、オーナー別一覧取得。公開設定（public/friends）の管理も行います。

`lib/repos/imageRepo.ts`

アルバム内画像の追加、削除、一覧取得。Storageへのアップロードも含みます。

`lib/repos/commentRepo.ts`

コメントの追加、削除、一覧取得。アルバム単位でのコメント取得をサポートします。

`lib/repos/likeRepo.ts`

いいねのトグル（追加/削除）、カウント、ユーザー一覧取得。いいね状態のチェック機能も提供します。

`lib/repos/repostRepo.ts`

リポストのトグル、カウント、リポストユーザー一覧取得。タイムライン表示用にアルバム単位のリポスト情報を取得します。

`lib/repos/reactionRepo.ts`

リアクションのトグル、一覧取得、絵文字別集計。リアクターアクション情報の取得も行います。

`lib/repos/friendRepo.ts`

フレンド申請の送信、承認、キャンセル、解除。受信済み申請の一覧、承認済みフレンドの一覧取得を提供します。

`lib/repos/watchRepo.ts`

ウォッチ（フォロー）の追加、削除、トグル。ウォッチ中オーナー ID 一覧、ウォッチャー一覧を取得します。

lib/repos/notificationRepo.ts

通知の追加、一覧取得、既読一括マーク。リアルタイム購読機能（onSnapshot）も提供します。

lib/repos/searchRepo.ts

ユーザー検索機能（handle/displayName の部分一致クエリ）。将来的にアルバム検索も拡張予定。

lib/repos/timelineRepo.ts

タイムライン用のアルバム取得クエリ。指定オーナー ID リストと公開設定を考慮してフェッチします。

src/ - アプリケーション層

src/hooks/ - カスタム React Hooks

src/hooks/useAuthUser.ts

Firebase Auth のユーザー状態を管理するフック。

src/hooks/useFriendship.ts

フレンド関係の取得・操作を行うフック。

src/hooks/useWatch.ts

ウォッч（フォロー）関係の取得・操作を行うフック。

src/hooks/useAlbumDetail.ts

アルバム詳細データの取得を行うフック。

src/hooks/useAlbumAccess.ts

アルバムへのアクセス権限をチェックするフック。

src/hooks/useNotificationsBadge.ts

未読通知数のバッジ表示用フック。

src/hooks/useTimelineItemVisibility.ts

タイムラインアイテムの可視性を検出するフック（IntersectionObserver 使用）。

src/hooks/useVerificationGuard.ts

メール確認済みかをチェックするガードフック。

src/hooks/useAsyncOperation.ts

非同期操作の loading/error 状態を管理する汎用フック。

src/hooks/useThumbBackfill.ts

サムネイル画像のバックフィル処理を行うフック。

src/services/ - ビジネスロジックサービス

src/services/avatar.ts

アバター画像のアップロードおよびリサイズ処理。react-easy-crop で生成した Blob を Storage へ保存します。

src/services/createAlbumWithImages.ts

アルバムと画像を一括作成するサービス。複数画像のアップロードをトランザクション的に処理します。

src/services/deleteAccount.ts

アカウント削除サービス。ユーザー情報、アルバム、画像、コメント、フレンド申請などを一括削除します（実装中）。

src/services/sendVerificationDev.ts

メール確認リンク送信サービス（開発用）。Nodemailer でメール送信を予定しています（実装中）。

src/services/timeline/listLatestAlbums.ts

タイムライン集約サービス。自分、フレンド、ウォッチ中ユーザーの最新アルバムを取得し、画像、コメントプレビュー、いいね、リポスト、リアクションを集約して TimelineItemVM を生成します。

src/services/profile/buildPatch.ts

プロフィール編集時のパッチデータを構築するサービス。

src/repositories/admin/ - Admin SDK リポジトリ

src/repositories/admin/firestore.ts

Admin SDK を使用する Firestore 操作関数（adminAddImage, adminDeleteImage, adminToggleLike, adminGetFriendStatus, adminGetAlbum, adminCanUploadMoreImages 等）。

src/libs/ - ライブラリラッパー

src/libs.firebaseioAdmin.ts

Firebase Admin SDK 初期化と認証・DB アクセス関数（verifyIdToken, getAdminAuth, getAdminDb） 。環境対応口ガードを使用。

src/models/

アプリケーション層のビュー・モデル定義（TimelineItemVM, UserRef 等）。

src/types/ - 型定義

src/types/firestore.ts

Firestore ドキュメント型を定義（ERR 定数を含む）。

src/types/firebase-admin.d.ts

Firebase Admin SDK の型定義。

app/album/[id]/hooks/ - アルバム詳細ページ用フック

app/album/[id]/hooks/useAlbumData.ts

アルバム、画像、コメント、リアクションのデータ取得とリアルタイム購読。

app/album/[id]/hooks/useLikes.ts

いいね状態の管理と楽観更新。

app/album/[id]/hooks/useReactions.ts

リアクション状態の管理、絵文字ピッカー、楽観更新。

app/album/[id]/hooks/useComments.ts

コメントの追加、編集、削除操作。

app/album/[id]/hooks/useAlbumEdit.ts

アルバムタイトル、公開設定の編集と保存。

app/album/[id]/hooks/useImageActions.ts

画像の追加、削除操作。

app/album/[id]/hooks/index.ts

フックの統一エクスポート。

app/user/[id]/hooks/ - プロフィールページ用フック

app/user/[id]/hooks/useProfileData.ts

プロフィールデータの取得と状態管理。

app/user/[id]/hooks/useProfileEdit.ts

プロフィール編集操作。

app/user/[id]/hooks/index.ts

フックの統一エクスポート。

types/ - 外部ライブラリ型定義

types/lightgallery-react.d.ts

lightgallery/react の型定義。

types/react-photo-album.d.ts

react-photo-album の型定義。

Firebase データモデル

コレクション一覧

- **users**: ユーザープロフィール情報 (uid, handle, displayName, bio, iconURL, coverURL, bannerURL)
 - **albums**: アルバム (ownerId, title, placeUrl, visibility, createdAt, updatedAt)
 - **albumImages**: アルバム内画像 (albumId, uploaderId, url, thumbUrl, createdAt)
 - **comments**: コメント (albumId, userId, body, createdAt)
 - **likes**: いいね (albumId, userId, createdAt) 。ドキュメント ID = `albumId_userId`
 - **reposts**: リポスト (albumId, userId, createdAt) 。ドキュメント ID = `albumId_userId`
 - **reactions**: リアクション (albumId, userId, emoji, createdAt) 。ドキュメント ID = `albumId:userId:emoji`
 - **friends**: フレンド申請 (userId, targetId, status, createdAt) 。ドキュメント ID = `userId_targetId`
 - **watches**: ウォッチ (userId, ownerId, createdAt) 。ドキュメント ID = `userId_ownerId`
 - **notifications**: 通知 (userId, actorId, type, albumId, message, createdAt, readAt)
 - **blocks**: ブロック (未実装)
-

Firebase セキュリティルール ([firestore.rules](#))

基本方針

- **認証チェック**: `isSignedIn()` でログインユーザーのみ書き込み可
- **所有者チェック**: `isUser(uid)` で自分の情報のみ更新可
- **フレンド関係チェック**: `isFriendWith(otherId)` で相互フレンド関係を確認

- **アルバム可視性チェック:** `canReadAlbum(albumId)` でアルバムが公開 (public) またはフレンド限定 (friends) の場合に読み取り許可
- **読み取りルール:** 基本的に **permissive (true)** で、クライアント側でフィルタリングを行う設計 (プロック解除後のリアルタイム更新に対応)
- **書き込みルール:** フレンド関係やアルバム可視性を厳密にチェックし、不正な作成を防止

ヘルパー関数

```
isSignedIn(): ログイン中か
isUser(uid): 自分のドキュメントか
isFriendWith(otherId): 双方向フレンド関係が accepted か
canReadAlbum(albumId): アルバムが public またはフレンド限定で権限あるか
```

各コレクションのルール概要

- **users:** read: 全員, create/update/delete: 本人のみ
- **albums:** read: 全員, create: ログインユーザーのみ, update/delete: オーナーのみ
- **albumImages:** read: 全員, create: アップローダー本人のみ (`canReadAlbum` チェック付き), delete: アップローダーまたはアルバムオーナー
- **comments:** read: 全員, create: ログインユーザー (`canReadAlbum` チェック付き), delete: コメント作成者またはアルバムオーナー
- **likes:** read: 全員, create: ログインユーザー (`canReadAlbum` チェック付き), delete: 本人のみ
- **reposts:** read: 全員, create: ログインユーザー (`canReadAlbum` チェック付き), delete: 本人のみ
- **reactions:** read: 全員, create: ログインユーザー (`canReadAlbum` チェック付き), delete: 本人のみ
- **friends:** read: ログインユーザーのみ, create: 本人が userId の申請のみ (status=pending), update: targetId 本人が承認操作のみ, delete: 申請者または対象者が削除可
- **watches:** read: ログインユーザーのみ, create: 本人が userId のウォッチのみ, delete: 本人のみ
- **notifications:** read: 本人のみ, create: actorId が本人の場合のみ, update: readAt を null → 非 null にする既読化のみ許可, delete: 不可
- **その他パス:** 全て閉じる (read: false, write: false)

フレンド限定機能

- アルバム visibility が '`friends`' の場合、オーナー本人または承認済みフレンドのみが `canReadAlbum` を通過
- comments, likes, reposts, reactions の create 時に `canReadAlbum` チェックが入るため、フレンド限定 アルバムへの操作は権限がないユーザーには不可

ブロック機能 (未実装)

- blocks コレクションのルールは未定義
- 将来的に `isBlockedWith(otherId)` ヘルパーと、インターフェイク制限ロジックを追加予定

機能一覧と実装ファイル対応表

1. 認証機能

- 実装ファイル:

- `lib/firebase.ts` (Firebase Auth 初期化)
- `components/AuthForm.tsx` (ログインフォーム)
- `components/AuthGate.tsx` (認証ガード)
- `app/login/page.tsx` (ログインページ)
- `app/register/page.tsx` (新規登録ページ)
- `app/register/complete/page.tsx` (登録完了ページ)
- `app/forgot-password/page.tsx` (パスワード忘れページ)
- `app/reset-password/page.tsx` (パスワードリセットページ)

- セキュリティルール: `users` コレクションで本人のみ `create/update` 可能

- 実装状況: 完了 (パスワードリセット機能を含む)

2. アルバム管理

- 実装ファイル:

- `lib/repos/albumRepo.ts` (アルバム CRUD)
- `app/album/[id]/page.tsx` (詳細・編集画面)
- `app/album/new/page.tsx` (新規作成画面)
- `components/AlbumCard.tsx` (カード UI)
- `components/album/AlbumActionsMenu.tsx` (操作メニュー)
- `lib/services/createAlbumWithImages.ts` (一括作成サービス)

- セキュリティルール: `albums` コレクション (オーナーのみ更新/削除可)

- 実装状況: 完了

3. 画像アップロード・管理

- 実装ファイル:

- `lib/repos/imageRepo.ts` (画像 CRUD)
- `components/upload/AlbumImageUploader.tsx` (アップローダー)
- `components/upload/AlbumImageCropper.tsx` (画像トリミング)
- `components/gallery/ImageGrid.tsx` (画像グリッド)
- `components/gallery/GalleryViewer.tsx` (lightgallery ビューアー)
- `app/api/images/add/route.ts` (画像追加 API)
- `app/api/images/delete/route.ts` (画像削除 API)

- セキュリティルール: `albumImages` コレクション (`uploaderId` 本人またはアルバムオーナーが削除可)

- 実装状況: 完了

4. タイムライン表示

- 実装ファイル:

- `app/timeline/page.tsx` (タイムライン画面)
- `src/services/timeline/listLatestAlbums.ts` (集約サービス)
- `lib/repos/timelineRepo.ts` (タイムライン用クエリ)
- `components/timeline/TimelineItem.tsx` (タイムライン 1 行 UI)

- 特徴: 自分、フレンド、ウォッチャユーザーの投稿を集約、無限スクロール、リアルタイム購読

- 実装状況: 完了

5. ソーシャル機能（フレンド・ウォッチ）

- **実装ファイル:**
 - `lib/repos/friendRepo.ts` (フレンド申請 CRUD)
 - `lib/repos/watchRepo.ts` (ウォッチ CRUD)
 - `components/profile/FriendActions.tsx` (フレンドボタン UI)
 - `components/profile/WatchActions.tsx` (ウォッチボタン UI)
 - `components/profile/FriendRemoveConfirmModal.tsx` (解除確認ダイアログ)
 - `app/user/[id]/page.tsx` (プロフィール画面で操作)
- **セキュリティルール:** friends/watches コレクションで本人のみ作成/削除可能
- **実装状況:** 完了

6. インタラクション（いいね・コメント・リポスト・リアクション）

- **実装ファイル:**
 - `lib/repos/likeRepo.ts` (いいね)
 - `lib/repos/commentRepo.ts` (コメント)
 - `lib/repos/repostRepo.ts` (リポスト)
 - `lib/repos/reactionRepo.ts` (リアクション)
 - `app/api/likes/toggle/route.ts` (いいねトグル API)
 - `app/api/reposts/toggle/route.ts` (リポストトグル API)
 - `app/api/reactions/toggle/route.ts` (リアクショントグル API)
 - `app/api/comments/add/route.ts` (コメント追加 API)
 - `components/comments/CommentItem.tsx, CommentInput.tsx` (コメント UI)
 - `components/timeline/ReactionSummary.tsx, ReactionPickerPopover.tsx` (リアクション UI)
- **セキュリティルール:** likes/reposts/reactions/comments で canReadAlbum チェック、本人のみ削除可能
- **実装状況:** 完了

7. 通知機能

- **実装ファイル:**
 - `lib/repos/notificationRepo.ts` (通知 CRUD、リアルタイム購読)
 - `app/notification/page.tsx` (通知一覧画面)
 - `components/icons/BellIcon.tsx` (通知アイコン)
- **セキュリティルール:** notifications コレクションで本人のみ読み取り、actorId 本人のみ作成、既読化のみ update 可能
- **実装状況:** 完了

8. 検索機能

- **実装ファイル:**
 - `lib/repos/searchRepo.ts` (ユーザー検索クエリ)
 - `app/search/page.tsx` (検索画面)
 - `components/icons/SearchIcon.tsx` (検索アイコン)
- **特徴:** handle/displayName の部分一致検索（インデックス要）
- **実装状況:** 完了（ユーザー検索のみ、アルバム検索は未実装）

9. ブロック機能（未実装）

- **実装予定ファイル:**
 - `lib/repos/blockRepo.ts` (現在空ファイル)
 - `firebase.rules` に `blocks` コレクション追加
 - `app/user/[id]/page.tsx` にブロック UI 追加
 - `src/services/timeline/listLatestAlbums.ts` でブロックユーザーフィルタリング追加
- **仕様:** Twitter 相当の相互ブロック、投稿非表示、コメント非表示、フレンド/ウォッチ強制解除
- **実装状況:** ✗ 未実装

10. プロフィール編集

- **実装ファイル:**
 - `app/user/[id]/page.tsx` (プロフィール画面でオンライン編集)
 - `lib/repos/userRepo.ts` (ユーザー情報更新)
 - `components/profile/AvatarModal.tsx` (アバター編集ダイアログ)
 - `components/profile/AvatarCropper.tsx` (画像トリミング UI)
 - `lib/services/avatar.ts` (アバターアップロードサービス)
- **特徴:** handle, displayName, bio, アイコン、カバー画像、バナー画像の編集に対応
- **実装状況:** ☑ 完了

11. アカウント削除

- **実装ファイル:**
 - `lib/services/deleteAccount.ts` (アカウント削除サービス、実装中)
 - `app/user/[id]/page.tsx` (削除フローと確認ダイアログ)
- **仕様:** ユーザー情報、アルバム、画像、コメント、フレンド申請などを一括削除（Storage も削除予定）
- **実装状況:** ◇ 実装中

12. パスワードリセット

- **実装ファイル:**
 - `app/forgot-password/page.tsx` (パスワード忘れページ)
 - `app/reset-password/page.tsx` (パスワードリセットページ)
 - Firebase Auth のパスワードリセット機能を使用
- **特徴:**
 - 第三者による変更防止（メールリンク経由のみ）
 - メールアドレス存在漏洩防止（中立的なメッセージ）
 - 今後の拡張: CAPTCHA、レート制限の追加
- **実装状況:** ☑ 基本機能完了（CAPTCHA/レート制限は未実装）

13. テーマ切り替え（未実装）

- **実装予定ファイル:**
 - `components/ThemeSwitch.tsx` (ライト/ダーク切り替えスイッチ)
 - Mantine の `useMantineColorScheme` で動的切り替え
- **現状:** ColorSchemeScript は導入済みだが、UI 未完成
- **実装状況:** ✗ 未実装

テスト

テストフレームワーク

- **Jest 29.7 + React Testing Library 16.3**
- **設定ファイル:** `jest.config.js, jest.setup.ts`

テストファイル

- `lib/repos/_tests_` - リポジトリ層のユニットテスト

テスト実行

```
npm run test      # すべてのテストを実行  
npm run test:watch # ウォッチモードで実行
```

実装状況

- テスト環境構築完了
- 一部リポジトリのテスト実装中
- E2E テスト未実装

未実装・実装中機能

未実装

1. **ロック機能:** blockRepo.ts は空ファイル、`firestore.rules` にもルール未定義
2. **テーマ切り替え UI:** ColorSchemeScript は準備済みだが、UI ボタンは非表示またはコメントアウト状態
3. **Discord Webhook 共有:** app/api/share/discord/route.ts はエンドポイントのみ存在、実装未完了
4. **アルバム通報機能:** app/api/reports/album/route.ts はエンドポイントのみ存在、モデレーション機能未実装
5. **アルバム検索:** searchRepo.ts はユーザー検索のみ実装、アルバム検索は未対応
6. **CAPTCHA/レート制限:** パスワードリセットに中立メッセージはあるが、CAPTCHA とレート制限は未実装
7. **E2E テスト:** Playwright または Cypress による E2E テスト環境未構築

実装中

1. **アカウント削除サービス:** deleteAccount.ts は部分実装、Storage 削除やトランザクション処理が未完成
2. **メール確認リンク送信:** sendVerificationDev.ts は Nodemailer 設定が未完成
3. **プロフィール画面の詳細:** app/user/[id]/page.tsx は機能豊富だが、参加アルバム・コメント・画像タブの集計ロジックが一部不完全
4. **ユニットテスト:** 一部リポジトリのテストは実装済みだが、カバレッジは不完全

パフォーマンス最適化

実装済みの最適化

1. **無限スクロール**: react-intersection-observer を使用したタイムラインの段階的読み込み
2. **画像の遅延読み込み**: lightgallery と react-photo-album による最適化
3. **リアルタイム購読の最適化**: 可視範囲のアイテムのみ購読（タイムライン）
4. **クエリの最適化**: Firestore インデックスの活用

今後の最適化課題

1. **N+1 問題の解消**: タイムライン表示時のバッチ取得実装
 2. **画像リサイズ**: Firebase Storage Extensions による自動サムネイル生成
 3. **キャッシュ戦略**: React Query の導入検討
 4. **コード分割**: Next.js の dynamic import 活用
-

開発・運用メモ

Firebase Emulator

- 開発時は Firebase Emulator Suite を使用してローカル環境でテスト可能
- `firebase emulators:start` でエミュレータを起動し、環境変数で接続先を切り替え

デプロイ

- Firestore ルール: `firebase deploy --only firestore:rules`
- Next.js アプリ: Vercel または Firebase Hosting へデプロイ

環境変数

- `NEXT_PUBLIC_FIREBASE_*`: Firebase クライアント SDK 用設定
- `FIREBASE_SERVICE_ACCOUNT_KEY`: Admin SDK 用サービスアカウント JSON (サーバー側)

依存関係更新

- `npm outdated` で最新バージョンを確認
- `npm update` で安全に更新
- メジャー・バージョンアップ時は互換性確認が必要

コードスタイル

- TypeScript strict モード有効
- ESLint でコードフォーマット統一 (`eslint.config.mjs`)
- コンポーネントは "use client" ディレクティブで明示的にクライアント化

テスト

- ユニットテスト: Jest 29.7 + React Testing Library 16.3 (導入済み)
- E2E テスト: Playwright または Cypress (未導入)
- テストコマンド: `npm run test`, `npm run test:watch`

今後の拡張予定

1. **ロック機能の完全実装**: blockRepo.ts の実装、firestore.rules への追加、タイムライン/プロフィールでのフィルタリング
2. **パスワードリセットの強化**: CAPTCHA、レート制限付きの安全な実装（基本機能は実装済み）
3. **テーマ切り替え UI の完成**: ライト/ダークモード切り替えボタンの実装
4. **アルバム検索機能**: タイトル・タグでの検索、フルテキスト検索の導入
5. **通報・モデレーション機能**: アルバム通報の処理フロー、管理者用ダッシュボード
6. **Discord Webhook 統合**: 投稿を Discord に自動共有する機能の完成
7. **通知プッシュ機能**: Firebase Cloud Messaging (FCM) でプッシュ通知
8. **リアクション絵文字のカスタマイズ**: ユーザーがカスタム絵文字を追加できる機能
9. **タグ機能**: アルバムにタグを付けて分類・検索できる機能
10. **統計・分析機能**: いいね数ランキング、人気アルバム表示など
11. **テストカバレッジの拡充**: E2E テスト環境構築、ユニットテストのカバレッジ向上

まとめ

本プロジェクトは、Next.js 16.1 App Router + Firebase を軸にしたフォト共有 SNS として、以下の機能を実装しています：

実装済み機能

- **認証機能**: メール/パスワード登録、ログイン、パスワードリセット
- **アルバム管理**: 作成、編集、削除、公開設定 (public/friends)
- **画像管理**: アップロード、トリミング、削除、ギャラリー表示
- **タイムライン**: 無限スクロール、リアルタイム更新
- **ソーシャル機能**: フренд申請/承認、ウォッチ機能
- **インタラクション**: いいね、コメント、リポスト、絵文字リアクション
- **通知機能**: リアルタイム通知、既読管理
- **検索機能**: ユーザー検索 (handle/displayName)
- **プロフィール**: 編集、アイコン/カバー/バナー画像設定
- **テスト環境**: Jest + React Testing Library

実装中機能

- **アカウント削除**: 一括削除ロジック (Storage 削除未完成)
- **ユニットテスト**: 一部リポジトリのテスト実装
- **プロフィールタブ**: 集計ロジックの一部改善

未実装機能

- **ロック機能**: リポジトリとルールが未実装
- **テーマ切り替え UI**: ColorScheme 準備済みだが UI 未完成
- **アルバム検索**: ユーザー検索のみ対応
- **Discord Webhook**: エンドポイントのみ存在
- **通報/モデレーション**: エンドポイントのみ存在
- **CAPTCHA/レート制限**: パスワードリセットの追加セキュリティ

- **E2E テスト:** テスト環境未構築

セキュリティ設計

Firestore セキュリティルールは、**読み取りを permissive**（基本的に true）にしてクライアント側でフィルタリングする設計です。これにより、ロック解除後のリアルタイム更新に柔軟に対応できます。**書き込み**は厳格にチェックし、フレンド関係やアルバム可視性を検証します。

パフォーマンス

- 無限スクロールによる段階的読み込み
- 可視範囲のみリアルタイム購読
- 画像の遅延読み込み
- Firestore インデックス最適化

今後は、ロック機能、アルバム検索、テーマ切り替え、通報機能などを段階的に実装予定です。

作成日: 2025-01-XX

最終更新: 2026-01-07

バージョン: 1.2

変更履歴

v1.2 (2026-01-07)

- **ディレクトリ構造の整理**
 - `lib/server/` ディレクトリを削除（未使用のre-exportファイル群）
 - `lib/errors.ts` を `lib/errors/index.ts` に統合
 - `lib/errors/USAGE_EXAMPLES.ts` を削除（実行されないサンプルコード）
 - `types/react-photo-gallery.d.ts` を削除（未使用）
 - `src/constants/` 空ディレクトリを削除
- **hooks のリファクタリング**
 - `lib/hooks/` を `src/hooks/` に移行
 - `lib/services/` を `src/services/` に移行
 - `app/album/[id]/page.tsx` を6つのカスタムフックに分割（885行→約270行）
 - `app/album/[id]/hooks/` ディレクトリを新規作成
 - `app/user/[id]/hooks/` ディレクトリを新規作成
- **Admin SDK 機能追加**
 - `adminGetFriendStatus`, `adminGetAlbum`, `adminCanUploadMoreImages` を追加
 - `/api/images/add` をAdmin SDK使用に変更（フレンド限定アルバムへの画像追加対応）

v1.1 (2026-01-02)

- 初期バージョン