App Overview

App Name: KELAB

Purpose: To generate and manage secure access codes for visitors entering an estate.

Target Users: Residents, visitors, and estate security personnel.

App Features

User Roles

1. **Resident**:
    - Register/Log in.
    - Generate access codes for visitors.
    - View and manage active/expired codes.
    - Receive notifications of visitor arrivals.
2. **Visitor**:
    - Input access code upon arrival.
    - Receive code via SMS, email, or a QR code.
3. **Security Personnel**:
    - Validate access codes.
    - Track visitor entry and exit.
4. **Admin**:
    - Manage estate user accounts.
    - Monitor app usage and security logs.

App Layout and Flow

1. Splash Screen

- **Purpose**: Brief welcome screen with the app logo.
- **Features**:
    - App name and tagline: *"Seamless Estate Access Made Simple."*
    - Loading animation.
    - Directs to login/register screen.

---

2. Authentication

- **Pages**:
    - **Login Page**:
        - Fields: Email/Phone, Password.
        - Buttons: Login, Forgot Password, Register.
    - **Registration Page**:
        - Fields: Name, Email, Phone Number, Password, Confirm Password.
        - User Type: Resident/Visitor/Security.
        - Verification via OTP.

---

3. Home Screen

- **Resident View**:
    - Generate Access Code button.
    - List of Active Access Codes (with status: Active/Used/Expired).
    - Quick Actions: Share Access Code, Revoke Access Code.
- **Visitor View**:
    - Field to Enter Access Code.
    - Option to scan a QR code.
    - Success/Failure messages for validation.
- **Security View**:
    - Search Access Code field.
    - Recent Entries log.
    - Entry/Exit Confirmation buttons.

---

4. Generate Access Code

- **Resident**:
    - Input visitor details: Name, Phone, Purpose, Duration of Visit.
    - Generate a random secure code (e.g., 6-digit alphanumeric).
    - Option to share via SMS, Email, or QR code.
    - Confirmation screen with code details.

---

5. Access Code Validation

- **Security View**:

    - Scan QR code or enter manually.
    - Verify visitor details: Code, Resident Name, Purpose.
    - Grant/Deny entry with logs updated in real-time.
- **Visitor View**:
    - Enter code and view status.
    - Confirmation message on successful entry.

---

6. Notifications

- **Resident**:
    - Push notifications: Visitor has arrived, access code used.
- **Visitor**:
    - Push/SMS notification: Access code details.

---

7. Settings

- Profile management for all users.
- Notification preferences.
- Security options: Enable two-factor authentication.

---

8. Admin Panel

- Separate web dashboard or app module for admins.
- Manage users, monitor logs, and generate reports.
- Manage app settings and access policies.

Technology Stack

Frontend:

- **Mobile App Framework**: Flutter or React Native (cross-platform support for iOS and Android).
- **UI Library**: Material Design or Ant Design.

Backend:

- **Language**: Node.js (Express.js), Python (Django/Flask), or Ruby on Rails.
- **Database**: PostgreSQL/MySQL for structured data, Firebase/Redis for real-time updates.

APIs:

- **Authentication**: Firebase Auth or OAuth2.
- **Messaging**: Twilio (SMS), SendGrid (email).
- **QR Code Generation**: Google Charts or a custom library.

Hosting:

- **Backend**: AWS, Google Cloud, or Azure.
- **Database**: Cloud SQL or DynamoDB.
- **Real-Time Data**: Firebase Realtime Database or WebSocket.

Security Considerations

1. Encrypted data storage for codes and personal information.
2. Secure code validation with expiry times.
3. Logging for all actions (code generation, validation, entry).
4. Two-factor authentication for residents and admins.