

# Mobile Apps

- [Modules](#)
  - [On Indonesia Sunshine - Complete Documentation](#)

# Modules

# On Indonesia Sunshine - Complete Documentation

“ **Application Name:** On Indonesia Sunshine  
**Type:** Mobile Attendance Management System  
**Version:** 1.0.0  
**Platform:** Android, iOS, Web, Desktop (Windows, macOS, Linux)  
**Last Updated:** February 2026

## Introduction

## About the Application

On Indonesia Sunshine is a comprehensive mobile attendance management system built with Flutter. The application enables employees to clock in and out with GPS verification, capture attendance photos, track their attendance history, and manage company assets.

## Key Features at a Glance

- ☐ GPS-based attendance tracking
- ☐ Photo verification for check-in/check-out
- ☐ Real-time location mapping
- ☐ Attendance history with calendar view
- ☐ Asset management system
- ☐ Role-based access control
- ☐ Secure authentication with JWT
- ☐ Multi-platform support

# Getting Started

## System Requirements

### For Development

- **Flutter SDK:** 3.1.0 or higher
- **Dart SDK:** Included with Flutter
- **IDE:** VS Code or Android Studio
- **OS:** macOS, Windows, or Linux

### For Android Development

- Android SDK (API 20+)
- Android Studio or command-line tools
- Java Development Kit (JDK)

### For iOS Development (macOS only)

- Xcode 12.0 or higher
- iOS SDK
- CocoaPods

## Installation

### Step 1: Install Flutter

#### macOS

```
brew install flutter
```

#### Windows

1. Download Flutter SDK from <https://flutter.dev>

2. Extract to C:\flutter
3. Add to PATH: C:\flutter\bin

## Linux

```
# Download and extract Flutter
cd ~/development
tar xf flutter_linux_*.tar.xz
export PATH="$PATH:`pwd`/flutter/bin"
```

## Step 2: Verify Installation

```
flutter doctor
```

## Step 3: Clone and Setup Project

```
# Clone repository
git clone <repository-url>
cd on-indonesia-absensi

# Install dependencies
flutter pub get

# Run the application
flutter run
```

---

# Architecture Overview

## Technology Stack

| Component | Technology     |
|-----------|----------------|
| Framework | Flutter 3.1.0+ |
| Language  | Dart           |

| Component        | Technology                    |
|------------------|-------------------------------|
| State Management | StatefulWidget + Global State |
| Networking       | HTTP Package                  |
| Local Storage    | SharedPreferences             |
| Maps             | Google Maps Flutter           |
| Authentication   | JWT (JSON Web Token)          |
| Backend API      | REST API                      |

# Application Architecture



# Project Structure

```

lib/
├── main.dart           # Application entry point

```

```
├─ services/           # Backend services
|  └─ onrest.dart      # API service (all HTTP calls)
|  └─ on_shared_preferences.dart # Local storage
|  └─ globals.dart     # Global state variables
└─ pages/             # UI screens
    └─ spalshscreen.dart # Splash screen
    └─ login/          # Login module
    └─ mainscreen.dart # Main navigation
    └─ camera_preview.dart # Camera functionality
    └─ screen/        # Feature screens
        └─ homescreen/ # Home & attendance
        └─ historyscreen/ # Attendance history
        └─ profilescreen/ # User profile
        └─ assetmanagement/ # Asset management
```

# Core Features

## 1. Authentication

### Login Process

The application uses JWT-based authentication:

1. User enters credentials (username/password)
2. App sends request to `/api/auth/login`
3. Server validates and returns JWT token
4. Token is stored in SharedPreferences
5. Token is included in all subsequent API requests

### Security Features

- JWT token-based authentication
- Secure storage using SharedPreferences
- Screenshot prevention (Android)
- Device identification tracking

- Jailbreak/Root detection

## Code Example: Login

```
Future loginApp(String username, String password) async {
  Uri url = Uri.parse('${apiAddress}/auth/login');

  final http.Response response = await http.post(
    url,
    headers: <String, String>{
      'Content-Type': 'application/json',
    },
    body: jsonEncode({
      "username": username,
      "password": password,
      "isNative": true
    }),
  );

  if (response.statusCode == 200) {
    var result = json.decode(response.body);
    // Store token and user info
    return result;
  }
}
```

## 2. Attendance Management

### Clock In Process

1. **Location Detection:** App gets current GPS coordinates
2. **Site Verification:** Finds nearest registered site
3. **Photo Capture:** Takes employee photo
4. **Validation:** Checks if within allowed distance
5. **Submission:** Sends data to server

# Clock Out Process

1. **Status Check:** Verifies user is clocked in
2. **Location Capture:** Gets current location
3. **Photo Verification:** Takes exit photo
4. **Update Record:** Updates existing attendance record

## Data Stored for Attendance

| Field                    | Description                |
|--------------------------|----------------------------|
| employeeCode             | Unique employee identifier |
| timeIn / timeOut         | Server timestamp           |
| locationIn / locationOut | GPS coordinates            |
| pictureIn / pictureOut   | Photo URLs                 |
| statusIn / statusOut     | Onsite/Offsite status      |
| offsiteReason            | Reason if offsite          |

## Attendance Status

- **Checked In:** Employee has clocked in, not yet out
- **Checked Out:** Employee has completed day
- **Offsite:** Working from non-registered location
- **Not Checked In:** No active attendance

## 3. Location Tracking

### GPS Integration

The app uses multiple location packages:

- **geolocator:** Gets current GPS coordinates
- **geocoding:** Converts coordinates to addresses
- **google\_maps\_flutter:** Displays maps

## Site Verification Algorithm

1. Get current location (latitude, longitude)
2. Fetch list of registered sites from API
3. Calculate distance to each site
4. Find nearest site
5. Check if distance < allowed radius
6. Set status: Onsite or Offsite

## Map Features

- Real-time location marker
- Nearby site markers
- Distance calculation
- Address display
- Refresh location button

# 4. History Tracking

## Features

- View attendance by month
- Calendar view with attendance dates
- Filter by date range
- Detailed record display
- Export functionality (future)

## History Data Display

For each attendance record:

- Date and time in/out
- Duration
- Location (address)
- Photos (in/out)
- Status (onsite/offsite)
- Approval status

# 5. Asset Management

“ **Note:** This module is role-based. Only users with asset management permission can access it.

## Asset Inventory

### Features:

- Add new assets
- Update asset information
- View asset details
- Track asset history
- Photo documentation
- Vendor management

### Asset Information:

- Asset number
- Asset name
- Serial number
- Brand and type
- Purchase date
- Price
- Condition
- Current location
- Assigned employee

## Asset Tracking

Track asset movements and maintenance:

- Service history
  - Location changes
  - Condition updates
  - Photo documentation
  - Employee assignments
-

# API Reference

## Base Configuration

Base URL: <https://sunshineapi.onindonesia.id/api>

Authentication: Bearer Token (JWT)

Content-Type: application/json

## Authentication Endpoints

### Login

POST /auth/login

Request Body:

```
{  
  "username": "string",  
  "password": "string",  
  "isNative": true  
}
```

Response (200):

```
{  
  "token": "jwt_token",  
  "name": "Employee Name",  
  "code": "EMP001",  
  "position": "Position Title"  
}
```

## Attendance Endpoints

# Create Attendance (Clock In)

POST /attendance/create

Authorization: Bearer {token}

Request Body:

```
{
  "employeeCode": "string",
  "locationIn": "address",
  "pictureIn": "base64_or_url",
  "latIn": "latitude",
  "longIn": "longitude",
  "statusIn": "onsite|offsite",
  "status": true,
  "timeIn": "timestamp",
  "offsiteReason": "reason_if_offsite",
  "dateImage": "date",
  "stage": "in"
}
```

Response (200):

```
{
  "id": "attendance_id",
  "message": "Success"
}
```

# Update Attendance (Clock Out)

POST /attendance/update

Authorization: Bearer {token}

Request Body:

```
{
  "id": "attendance_id",
  "employeeCode": "string",
  "locationOut": "address",
  "pictureOut": "base64_or_url",
  "latOut": "latitude",
}
```

```
"longOut": "longitude",
"statusOut": "onsite|offsite",
"status": true,
"timeOut": "timestamp",
"offsiteReason": "reason_if_offsite",
"dateImage": "date",
"stage": "out"
}
```

## Check Attendance Status

```
GET /attendance/check?employeeCode={code}
```

```
Authorization: Bearer {token}
```

Response (200):

```
{
  "status": "checked_in|checked_out|none",
  "now": {
    "in": {
      "timeIn": "timestamp",
      "locationIn": "address",
      ...
    },
    "out": { ... }
  }
}
```

## Get Attendance History

```
POST /attendance/personal
```

```
Authorization: Bearer {token}
```

Request Body:

```
{
  "employeeCode": "string",
  "date": "YYYY-MM"
}
```

Response (200):

```
[
  {
    "id": "id",
    "date": "date",
    "timeIn": "time",
    "timeOut": "time",
    "locationIn": "address",
    "locationOut": "address",
    ...
  }
]
```

## Upload Image

POST /attendance/upload-image

Authorization: Bearer {token}

Content-Type: multipart/form-data

Form Data:

- image: File
- employeeName: string
- statement: string

Response (200):

```
{
  "url": "image_url",
  "message": "Success"
}
```

## Location Endpoints

### Get Server Time

```
GET /servertime?isDate=true
```

Response (200):

```
{  
  "server_time": "timestamp",  
  "date": "YYYY-MM-DD"  
}
```

## Get Site List

```
GET /attendance/get-site-info
```

Authorization: Bearer {token}

Response (200):

```
[  
  {  
    "name": "Site Name",  
    "latitude": "lat",  
    "longitude": "long",  
    "radius": "meters"  
  }  
]
```

## Get Nearest Site

```
POST /attendance/get-nearest-site
```

Authorization: Bearer {token}

Request Body:

```
{  
  "latitude": "current_lat",  
  "longitude": "current_long"  
}
```

Response (200):

```
{  
  "site": "Site Name",  
  "distance": "meters",  
}
```

```
"isWithinRadius": true
}
```

# User Endpoints

## Change Password

POST /user/change-password

Authorization: Bearer {token}

Request Body:

```
{
  "oldPassword": "string",
  "newPassword": "string"
}
```

Response (200):

```
{
  "message": "Password updated successfully"
}
```

## Check User Access Menu

GET /user/accessmenu

Authorization: Bearer {token}

Response (200):

```
{
  "assetManagement": "true|false",
  "otherPermissions": "..."
}
```

# Asset Management Endpoints

# Get Company List

GET /assetManagement/company

Authorization: Bearer {token}

Response (200):

```
[
  {
    "id": "company_id",
    "name": "Company Name"
  }
]
```

# Get Asset Inventory

GET /assetManagement/inventories?company={name}&filter={field}&key={value}

Authorization: Bearer {token}

Response (200):

```
[
  {
    "assetNumber": "AST001",
    "assetName": "Laptop",
    "serialNumber": "SN123",
    "status": "active",
    ...
  }
]
```

# Add Asset

POST /assetManagement/inventories-mobile

Authorization: Bearer {token}

Content-Type: multipart/form-data

Form Data:

- assetPicture: File[] (multiple)
- employeePicture: File
- assetData: JSON string

Response (201):

```
{  
  "message": "Asset created successfully",  
  "assetNumber": "AST001"  
}
```

## Update Asset

PUT /assetManagement/inventories

Authorization: Bearer {token}

Request Body:

```
{  
  "assetNumber": "AST001",  
  "assetName": "Updated Name",  
  "serialNumber": "SN123",  
  "vendor": "Vendor Name"  
}
```

Response (200):

```
{  
  "message": "Asset updated successfully"  
}
```

## Get Asset Details

GET /assetManagement/tracking?assetNumber={number}

Authorization: Bearer {token}

Response (200):

```
{  
  "asset": { ... },  
  "history": [ ... ]  
}
```

---

# User Interface Guide

## Navigation Structure

### Bottom Navigation Tabs

1. **Home** (🏠) - Attendance and location
2. **History** (📅) - Attendance history
3. **Profile** (👤) - User profile and settings
4. **Assets** (📁) - Asset management (if authorized)

## Screen Descriptions

### Splash Screen

- First screen shown on app launch
- Displays app logo
- Checks authentication status
- Redirects to Login or Home

### Login Screen

#### Elements:

- Email/Username input field
- Password input field (with show/hide toggle)
- Login button
- Error message display

#### Validation:

- Required fields check
- Network connectivity check
- Credential validation

# Home Screen

## **Top Section:** Employee Profile Card

- Profile photo
- Employee name
- Position
- Employee code

## **Location Section:** Map View

- Current location marker
- Nearby site markers
- Distance indicator
- Refresh location button

**Instruction Banner:** "Press the  button to find position, and Refresh button to place onsite position"

## **Attendance Section:** Clock In/Out Card

- Current time display
- Status indicator (In/Out)
- Check In button (when available)
- Check Out button (when checked in)
- Offsite reason input (if not in site radius)

## **Last Attendance Section:** Recent Record

- Date
- Time In/Out
- Location
- Status
- Photos

# History Screen

## **Calendar View:**

- Month selector
- Calendar grid
- Highlighted attendance dates
- Date selection

## **Filter Options:**

- Date range picker
- Status filter (all, complete, incomplete)

#### **List View:**

- Chronological list of records
- Each item shows:
  - Date
  - Time in/out
  - Duration
  - Location
  - Status badge

#### **Detail View** (tap on record):

- Full attendance details
- Maps showing in/out locations
- Photos (in/out)
- Timestamps
- Status information

## Profile Screen

#### **Profile Information:**

- Profile photo
- Full name
- Employee code
- Position
- Department
- Company

#### **Actions:**

- Change Password button
- Logout button

## Change Password Screen

#### **Form Fields:**

- Current password
- New password
- Confirm new password

## Validation:

- Current password verification
- Password strength check
- Confirmation match check

# Asset Management Screens

## Asset Menu

- Asset Management option
- Asset Inventory option

## Asset List

### Features:

- Search bar
- Company filter
- Asset cards showing:
  - Photo
  - Asset name
  - Asset number
  - Status badge

## Asset Detail View

### Tabs:

1. **Information**
  - Asset details
  - Owner information
  - Purchase details
2. **History**
  - Service records
  - Location changes
  - Assignments
3. **Photos**
  - Gallery view
  - Photo details

## Add/Edit Asset

### Form Sections:

- Basic Information
  - Asset Details
  - Owner Information
  - Vendor Details
  - Purchase Information
  - Photo Uploads
- 

# Development Guide

## Project Setup for Developers

### 1. Environment Setup

```
# Install Flutter
flutter doctor

# Clone repository
git clone <repo-url>
cd on-indonesia-absensi

# Install dependencies
flutter pub get

# Verify setup
flutter analyze
```

### 2. Running the Application

```
# List available devices
flutter devices

# Run on connected device
flutter run
```

```
# Run on specific device
```

```
flutter run -d <device-id>
```

```
# Run on emulator
```

```
flutter run -d emulator-5554
```

## 3. Development Workflow

**Hot Reload:** Press `r` in terminal

- Quick UI updates
- Preserves app state

**Hot Restart:** Press `R` in terminal

- Full app restart
- Clears state

**Stop App:** Press `q` in terminal

## Code Structure

### Services Layer

OnRest ( `lib/services/onrest.dart` )

Singleton class handling all API communication.

**Usage:**

```
// Get instance
var api = OnRest.instance;

// Make API call
var result = await api.loginApp(username, password);

// Check result
if (result != 'error') {
  // Success
```

```
} else {  
  // Handle error  
}
```

## OnSharedPreferences (lib/services/on\_shared\_preferences.dart)

Wrapper for SharedPreferences.

### Usage:

```
// Save data  
await OnSharedPreferences.instance.setStringValue("key", "value");  
  
// Load data  
String value = await OnSharedPreferences.instance.getStringValue("key");  
  
// Delete data  
await OnSharedPreferences.instance.removeValue("key");  
  
// Clear all  
await OnSharedPreferences.instance.clear();
```

## Globals (lib/services/globals.dart)

Global state variables.

### Common Variables:

```
import 'package:on_indonesia_sunshine/services/globals.dart' as globals;  
  
// Authentication  
globals.jwt      // JWT token  
globals.code     // Employee code  
  
// Attendance  
globals.statusattendance // Current status  
globals.timeIn    // Clock in time  
globals.timeOut  // Clock out time  
globals.capturedImage // Photo file
```

```
// Location
globals.latIn, globals.longIn
globals.latOut, globals.longOut
globals.locationIn, globals.locationOut
```

# Adding New Features

## Example: Adding a New Screen

### 1. Create Screen File

```
// lib/pages/screen/myfeature/my_screen.dart
import 'package:flutter/material.dart';

class MyScreen extends StatefulWidget {
  const MyScreen({super.key});

  @override
  State<MyScreen> createState() => _MyScreenState();
}

class _MyScreenState extends State<MyScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('My Feature')),
      body: Center(child: Text('Content')),
    );
  }
}
```

### 2. Add Route

```
// In main.dart
routes: {
  '/myscreen': (context) => const MyScreen(),
  // ...
}
```

```
},
```

### 3. Navigate to Screen

```
Navigator.pushNamed(context, '/myscreen');
```

## Example: Adding API Endpoint

```
// In lib/services/onrest.dart

Future<dynamic> myNewEndpoint(String param) async {
  final jwt = await OnSharedPreferences.instance.getStringValue("accessToken");
  Uri url = Uri.parse('$apiAddress/my-endpoint');

  final http.Response response = await http.post(
    url,
    headers: <String, String>{
      'Content-Type': 'application/json',
      HttpHeaders.authorizationHeader: "Bearer $jwt"
    },
    body: jsonEncode(<String, String>{
      "parameter": param,
    }),
  );

  if (response.statusCode == 200) {
    if (kDebugMode) {
      print(response.body);
    }
    return jsonDecode(response.body);
  } else {
    return {'error': 'Request failed'};
  }
}
```

## Building for Production

# Android Build

```
# 1. Update version in pubspec.yaml
version: 1.0.1+2

# 2. Build release APK
flutter build apk --release

# 3. Or build App Bundle (for Play Store)
flutter build appbundle --release

# Output locations:
# APK: build/app/outputs/flutter-apk/app-release.apk
# AAB: build/app/outputs/bundle/release/app-release.aab
```

# iOS Build

```
# 1. Update version
# Edit pubspec.yaml

# 2. Open in Xcode
open ios/Runner.xcworkspace

# 3. In Xcode:
# - Select "Any iOS Device"
# - Product → Archive
# - Distribute App
```

# Build Checklist

- Update version number
- Test on physical devices
- Verify production API URL
- Remove debug code
- Run `flutter analyze`
- Test critical flows

# Configuration

## Google Maps API Setup

### 1. Get API Key

1. Go to [Google Cloud Console](#)
2. Create or select project
3. Enable Maps SDK for Android
4. Enable Maps SDK for iOS
5. Create API Key

### 2. Configure Android

Edit `android/app/src/main/AndroidManifest.xml`:

```
<meta-data
  android:name="com.google.android.geo.API_KEY"
  android:value="YOUR_API_KEY_HERE"/>
```

### 3. Configure iOS

Edit `ios/Runner/AppDelegate.swift`:

```
import GoogleMaps

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
  override func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?)
```

```
) -> Bool {  
  GMSServices.provideAPIKey("YOUR_API_KEY_HERE")  
  GeneratedPluginRegistrant.register(with: self)  
  return super.application(application, didFinishLaunchingWithOptions: launchOptions)  
}  
}
```

# Environment Configuration

Currently using hardcoded API URL. To support multiple environments:

## Create Config File

```
// lib/config/app_config.dart  
class AppConfig {  
  static const String apiUrl = String.fromEnvironment(  
    'API_URL',  
    defaultValue: 'https://sunshineapi.onindonesia.id/api',  
  );  
  
  static const String env = String.fromEnvironment(  
    'ENV',  
    defaultValue: 'production',  
  );  
}
```

## Use in Code

```
import 'package:on_indonesia_sunshine/config/app_config.dart';  
  
String apiAddress = AppConfig.apiUrl;
```

## Build with Environment

```
# Development
```

```
flutter run --dart-define=API_URL=https://dev.api.com --dart-define=ENV=development
```

```
# Staging
```

```
flutter build apk --dart-define=API_URL=https://staging.api.com --dart-define=ENV=staging
```

```
# Production
```

```
flutter build apk --dart-define=API_URL=https://sunshineapi.onindonesia.id/api --dart-define=ENV=production
```

# Troubleshooting

## Common Issues

### 1. "Flutter SDK not found"

```
# Verify Flutter installation
```

```
flutter doctor
```

```
# Add Flutter to PATH
```

```
export PATH="$PATH:[PATH_TO_FLUTTER]/flutter/bin"
```

### 2. "Waiting for another flutter command"

```
# Remove lock file
```

```
rm -rf ~/.flutter/flutter.lock
```

### 3. "CocoaPods not installed" (iOS)

```
# Install CocoaPods
```

```
sudo gem install cocoapods
```

```
# Install pods
```

```
cd ios
pod install
```

## 4. Map not showing

- Verify Google Maps API key
- Check API key restrictions
- Enable Maps SDK in Google Cloud Console
- Check internet connectivity

## 5. Location permission denied

```
// Request permission in code
import 'package:permission_handler/permission_handler.dart';

var status = await Permission.location.request();
if (status.isGranted) {
  // Permission granted
} else {
  // Handle permission denied
}
```

## 6. Build errors after update

```
flutter clean
flutter pub get
cd ios && pod install && cd .. # For iOS
flutter run
```

## 7. Hot reload not working

- Press `R` for hot restart
  - Stop and restart app
  - Check for syntax errors
-

# Security Best Practices

## Application Security

### 1. Screenshot Prevention

```
// Enabled in main.dart for Android
await FlutterWindowManager.addFlags(
  FlutterWindowManager.FLAG_SECURE
);
```

### 2. Secure Storage

- JWT tokens stored in SharedPreferences
- Never log sensitive data in production
- Clear data on logout

### 3. API Security

- All requests use HTTPS
- JWT token in Authorization header
- Token expiration handling

### 4. Location Privacy

- Request permissions properly
- Only collect when needed
- Clear explanation to users

### 5. Code Security

- No hardcoded credentials
- API keys not in version control

- Obfuscate code for release builds

# Release Security Checklist

- Remove all debug prints
  - Verify API endpoints (HTTPS)
  - Check for exposed API keys
  - Enable code obfuscation
  - Test on rooted/jailbroken devices
  - Review permissions
  - Secure local storage
- 

## Maintenance

### Regular Maintenance Tasks

#### Monthly

- Update Flutter SDK
- Update dependencies
- Review security advisories
- Check API changes

#### Before Each Release

- Update version number
- Run full test suite
- Update documentation
- Review performance

Check error logs

# Updating Dependencies

```
# Check for updates
flutter pub outdated

# Update to latest compatible versions
flutter pub upgrade

# Update to major versions (careful!)
flutter pub upgrade --major-versions

# After updating
flutter clean
flutter pub get
flutter run
```

# Version Management

## Version Format

```
# In pubspec.yaml
version: 1.2.3+45

# 1.2.3 = Version Name (MAJOR.MINOR.PATCH)
# 45 = Version Code (build number)
```

# Semantic Versioning

- **MAJOR:** Breaking changes
  - **MINOR:** New features (backward compatible)
  - **PATCH:** Bug fixes
-

# Appendix

## Dependencies Reference

### UI & Navigation

- `google_nav_bar` - Bottom navigation
- `salomon_bottom_bar` - Alternative nav bar
- `animated_notch_bottom_bar` - Animated nav
- `line_icons` - Icon pack
- `page_transition` - Page animations

### Network & Storage

- `http` - HTTP client
- `shared_preferences` - Local storage
- `url_launcher` - Launch URLs

### Location & Maps

- `google_maps_flutter` - Maps
- `location` - Location services
- `geolocator` - Geolocation
- `geocoding` - Reverse geocoding

### Media

- `image_picker` - Camera/Gallery
- `path` - File paths

### Date & Time

- `intl` - Formatting
- `slide_digital_clock` - Clock widget
- `scrollable_clean_calendar` - Calendar

- [easy\\_date\\_timeline](#) - Timeline

## Security

- [flutter\\_jailbreak\\_detection](#) - Root detection
- [permission\\_handler](#) - Permissions
- [flutter\\_windowmanager](#) - Screenshot prevention

## Glossary

| Term       | Definition                                    |
|------------|---|
| JWT        | JSON Web Token - Authentication token         |
| GPS        | Global Positioning System - Location tracking |
| API        | Application Programming Interface             |
| SDK        | Software Development Kit                      |
| Onsite     | Employee at registered location               |
| Offsite    | Employee at non-registered location           |
| Clock In   | Mark arrival/start of work                    |
| Clock Out  | Mark departure/end of work                    |
| Asset      | Company equipment or property                 |
| Geofencing | Location-based boundaries                     |

## Support Resources

### Official Documentation

- [Flutter Docs](#)
- [Dart Language](#)
- [Pub.dev Packages](#)

## Community

- [Flutter Community](#)
  - [Stack Overflow](#)
  - [GitHub Issues](#)
- 

**Document Version:** 1.0

**Last Updated:** February 2026

**Maintained By:** On Indonesia Development Team