Contents

Privacy-Focused Context-Aware Digital Wellbeing System - Complete Code Documentation

Version: 1.0.0
Generated: 2026-01-02T14:15:56.156242
Total Files: 128
Total Lines: 35,452

Table of Contents

Project Overview

Privacy-Focused Context-Aware Digital Wellbeing System is a comprehensive solution that combines backend API, mobile application, AI/ML models, and IoT integration to provide intelligent notification management, focus mode, privacy protection, and wellness monitoring.

Statistics

- **Total Files Analyzed:** 128
- **Total Lines of Code:** 35,452

Files by Type

- **Python:** 76 files
- **Javascript:** 44 files

- **Json:** 8 files

Files by Category

- **Iot:** 7 files
- **Configs:** 2 files
- **Ai Models:** 13 files
- **Ai Tests:** 3 files
- **Backend:** 37 files
- **Backend Tests:** 19 files
- **Mobile:** 40 files
- **Mobile Tests:** 7 files

Backend API

The backend API is built with FastAPI and provides RESTful endpoints for all system functionality.

API Endpoints

backend-api/app/api/init.py **Lines:** 1
Type: Python Module

Description: API routes package...

backend-api/app/api/ai_advanced.py **Lines:** 385
Type: Python Module

Description: Advanced AI API Endpoints Provides endpoints for priority scoring, focus prediction, suggestions, and behavior analysis...

Classes:

- `NotificationPriorityRequest` (line 33)
- `NotificationPriorityResponse` (line 38)
- `FocusPredictionRequest` (line 44)
- `FocusPredictionResponse` (line 52)
- `DailyFocusScheduleResponse` (line 59)
- `SuggestionRequest` (line 63)
- `SuggestionResponse` (line 78)
- `BehaviorInsightsResponse` (line 86) **Key Imports:** 14 modules

backend-api/app/api/analytics.py **Lines:** 811
Type: Python Module

Description: Analytics API Endpoints Comprehensive user analytics and insights endpoints...

Classes:

- `PeriodType` (line 22)
  - Time period types
- `SessionTrackRequest` (line 30)
  - Track a user session
- `ScreenTimeTrackRequest` (line 38)
  - Track screen time
- `FocusSessionTrackRequest` (line 47)
  - Track focus session
- `BreakTrackRequest` (line 56)
  - Track break
- `NotificationTrackRequest` (line 64)
  - Track notification
- `DistractionTrackRequest` (line 73)

  - Track distraction
- `GoalSetRequest` (line 82)
  - Set a goal
- `GoalUpdateRequest` (line 91)
  - Update goal progress **Key Imports:** 15 modules

backend-api/app/api/auth.py Lines: 97
Type: Python Module

Description: Authentication API endpoints Handles user registration, login, and token management...

Classes:

- `UserRegister` (line 14)
- `UserLogin` (line 19)
- `TokenResponse` (line 23)
- `UserResponse` (line 29) **Key Imports:** 9 modules

backend-api/app/api/devices.py Lines: 150
Type: Python Module

Description: Devices API endpoints Handles IoT device registration and management...

Classes:

- `DeviceRegister` (line 14)
- `DeviceResponse` (line 19)
- `DeviceCommand` (line 25)
- `DeviceInfo` (line 29) **Key Imports:** 8 modules

backend-api/app/api/iot_automation.py Lines: 203
Type: Python Module

Description: IoT Automation API Endpoints Handles automated responses to sensor data and smart environment controls...

Classes:

- `SensorDataInput` (line 18)
- `FocusModeSchedule` (line 27)
- `ThresholdConfig` (line 33) **Key Imports:** 11 modules

backend-api/app/api/notifications.py Lines: 130
Type: Python Module

Description: Notifications API endpoints Handles notification classification and management...

Classes:

- `NotificationClassify` (line 15)
- `ClassificationResponse` (line 20)
- `NotificationItem` (line 26)
- `NotificationListResponse` (line 33) **Key Imports:** 8 modules

backend-api/app/api/privacy.py Lines: 109
Type: Python Module

Description: Privacy API endpoints Handles VPN, caller ID masking, and privacy features...

Classes:

- `VPNStatusResponse` (line 13)
- `PrivacyStatusResponse` (line 18) **Key Imports:** 3 modules

backend-api/app/api/privacy_advanced.py **Lines:** 489
Type: Python Module

Description: Advanced Privacy API Endpoints Provides API for VPN, caller masking, location spoofing, and network security...

Classes:

- `VPNConnectRequest` (line 23)
- `CallScreenRequest` (line 28)
- `BlockNumberRequest` (line 33)
- `ReportSpamRequest` (line 37)
- `SetLocationRequest` (line 42)
- `SetLocationModeRequest` (line 47)
- `BlockDomainRequest` (line 51) **Key Imports:** 18 modules

backend-api/app/api/wellbeing.py **Lines:** 151
Type: Python Module

Description: Wellbeing API endpoints Handles focus mode, productivity tracking, and analytics...

Classes:

- `FocusModeRequest` (line 14)
- `FocusModeResponse` (line 19)
- `WellbeingStats` (line 25) **Key Imports:** 6 modules

Services

backend-api/app/services/init.py **Lines:** 1
Type: Python Module

Description: Services package...

backend-api/app/services/analytics_tracker.py **Lines:** 543
Type: Python Module

Description: User Analytics Tracker Comprehensive analytics service for tracking user behavior and generating insights...

Classes:

- `AnalyticsTracker` (line 13)
  - Track and analyze user behavior patterns for data-driven insights
  - Methods: 17
    - `__init__(self)`
    - `track_session(self, user_id, start_time, end_time, device_type)`
    - `track_screen_time(self, user_id, app_name, duration_minutes, timestamp, category)`
    - `track_focus_session(self, user_id, start_time, end_time, quality_score, task_name)`
    - `track_break(self, user_id, start_time, duration_minutes, break_type)` **Key Imports:** 9 modules

backend-api/app/services/caller_masking.py **Lines:** 271
Type: Python Module

Description: Caller ID Masking Service Blocks spam calls, masks caller information, and provides call screening...

Classes:

- `CallType` (line 16)
- `CallerIDMasking` (line 25)
  - Manage caller ID masking and spam detection
  - Methods: 4

        * `__init__(self)`
        * `_init_spam_database(self)`
        * `_normalize_phone_number(self, phone_number)`
        * `_check_spam_database(self, phone_number)` **Key Imports:** 8 modules

backend-api/app/services/dnd_scheduler.py **Lines:** 381
Type: Python Module

Description: Do Not Disturb (DND) Scheduler Manages automated DND schedules and rules...

Classes:

- `DNDScheduleType` (line 12)
  - Types of DND schedules
- `DNDException` (line 20)
  - Exceptions to DND rules
- `DNDScheduler` (line 29)
  - Manage Do Not Disturb schedules and automation
  - Methods: 19
    * `__init__(self)`
    * `create_schedule(self, user_id, schedule_type, start_time, end_time, days_of_week, enabled, exceptions)`
    * `get_user_schedules(self, user_id)`
    * `get_schedules(self, user_id)`
    * `update_schedule(self, user_id, schedule_id, updates)` **Key Imports:** 8 modules

backend-api/app/services/insights_generator.py **Lines:** 499
Type: Python Module

Description: Insights Generator Advanced insights generation using pattern recognition and ML predictions...

Classes:

- `InsightsGenerator` (line 12)
  - Generate advanced insights from user analytics data
  - Methods: 15
    * `__init__(self)`
    * `identify_peak_hours(self, hourly_data)`
    * `detect_usage_patterns(self, daily_summaries)`
    * `predict_optimal_schedule(self, weekly_data)`
    * `generate_personalized_tips(self, insights_data, user_patterns)` **Key Imports:** 8 modules

backend-api/app/services/iot_automation.py **Lines:** 371
Type: Python Module

Description: IoT Automation Service Handles automated responses to sensor data and environmental conditions...

Classes:

- `AutomationType` (line 14)
- `AlertSeverity` (line 22)
- `IoTAutomationService` (line 29)
  - Manages automated responses to IoT sensor data
  - Methods: 2
    * `__init__(self)`
    * `_log_automation(self, automation)` **Key Imports:** 7 modules

backend-api/app/services/location_spoofing.py **Lines:** 275
Type: Python Module

Description: Location Spoofing Service Provides location privacy by spoofing GPS coordinates...

Classes:

- `LocationMode` (line 16)
- `LocationSpoofing` (line 23)
  - Manage location spoofing and privacy
  - Methods: 2
    - `__init__(self)`
    - `_get_mode_description(self, mode)` **Key Imports:** 9 modules

backend-api/app/services/ml_model_service.py **Lines:** 459
Type: Python Module

Description: ML Model Service - Production ML Model Integration Handles loading, versioning, caching, and inference for notification classification...

Classes:

- `ModelCache` (line 22)
  - LRU cache for model predictions to reduce inference time
  - Methods: 6
    - `__init__(self, max_size, ttl_seconds)`
    - `_generate_key(self, text, sender)`
    - `get(self, text, sender)`
    - `set(self, text, sender, prediction)`
    - `clear(self)`
- `ModelVersionManager` (line 92)
  - Manages multiple model versions with rollback capability
  - Methods: 8
    - `__init__(self, models_dir)`
    - `_load_versions(self)`
    - `_save_versions(self)`
    - `register_version(self, version, metadata)`
    - `set_current_version(self, version)`
- `MLModelService` (line 174)
  - Production ML Model Service with versioning, caching, and monitoring
  - Methods: 10
    - `__init__(self, models_dir)`
    - `_load_model(self, version)`
    - `classify(self, text, sender, received_at, use_cache)`
    - `_determine_action(self, is_urgent, confidence)`
    - `_generate_reasoning(self, text, is_urgent, confidence)` **Functions:**
- `get_ml_service()` (line 454)
  - Get singleton ML service instance **Key Imports:** 14 modules

backend-api/app/services/mqtt_service.py **Lines:** 172
Type: Python Module

Description: MQTT Service Handles pub/sub messaging for real-time communication...

Classes:

- `MQTTService` (line 13)
  - MQTT client for real-time messaging
  - Methods: 13
    - `__init__(self, broker_host, broker_port)`
    - `on_connect(self, client, userdata, flags, rc)`
    - `on_disconnect(self, client, userdata, rc)`
    - `on_message(self, client, userdata, msg)`
    - `connect(self, username, password)` **Functions:**
- `get_mqtt_service()` (line 170)

  - Get global MQTT service instance **Key Imports:** 5 modules

backend-api/app/services/network_monitor.py **Lines:** 356
Type: Python Module

Description: Network Security Monitor Monitors network traffic, detects threats, and provides security insights...

Classes:

- `ThreatLevel` (line 17)
- `ThreatType` (line 25)
- `NetworkSecurityMonitor` (line 35)
  - Monitor network security and detect threats
  - Methods: 2
    * `__init__(self)`
    * `_get_threat_description(self, threat_type)` **Key Imports:** 10 modules

backend-api/app/services/notification_bundler.py **Lines:** 377
Type: Python Module

Description: Notification Bundling Service Groups similar notifications into digestible bundles...

Classes:

- `BundleType` (line 13)
  - Types of notification bundles
- `BundleStrategy` (line 22)
  - How to create bundles
- `NotificationBundler` (line 29)
  - Bundle notifications intelligently to reduce interruptions
  - Methods: 14
    * `__init__(self)`
    * `add_to_bundle(self, user_id, notification, bundle_strategy)`
    * `get_bundle(self, user_id, bundle_key, clear_after)`
    * `get_ready_bundles(self, user_id)`
    * `get_all_bundles(self, user_id)` **Key Imports:** 8 modules

backend-api/app/services/notification_filter.py **Lines:** 369
Type: Python Module

Description: Context-Aware Notification Filter Intelligently filters notifications based on user context, time, and behavior patterns...

Classes:

- `NotificationContext` (line 12)
  - User context states
- `FilterAction` (line 23)
  - Actions to take on notifications
- `NotificationPriority` (line 32)
  - Notification priority levels
- `ContextAwareFilter` (line 41)
  - Filter notifications based on user context and intelligent rules
  - Methods: 14
    * `__init__(self)`
    * `analyze_notification(self, notification_text, sender, timestamp, app_name, user_id)`
    * `_determine_priority(self, text, sender, app_name)`
    * `_get_user_context(self, user_id, timestamp)`
    * `_decide_action(self, priority, context, timestamp, app_name, user_id)` **Key Imports:** 10 modules

backend-api/app/services/notification_queue.py **Lines:** 385
Type: Python Module

Description: Priority-Based Notification Queue Manages intelligent queuing and batching of notifications...

Classes:

- `QueuePriority` (line 13)
  - Priority levels for queue (lower number = higher priority)
- `DeliveryStrategy` (line 22)
  - How to deliver queued notifications
- `NotificationQueue` (line 30)
  - Priority queue for notifications with intelligent batching
  - Methods: 17
    - `__init__(self)`
    - `enqueue(self, user_id, notification, priority, delivery_strategy)`
    - `dequeue(self, user_id, count)`
    - `peek(self, user_id, count)`
    - `cancel(self, user_id, queue_id)` **Key Imports:** 9 modules

backend-api/app/services/optimized_analytics.py **Lines:** 216
Type: Python Module

Description: Optimized Analytics Data Aggregation Service Pre-computed metrics and efficient data queries...

Classes:

- `OptimizedAnalyticsService` (line 15)
  - Provides pre-aggregated analytics data for fast retrieval
  - Methods: 3
    - `__init__(self)`
    - `_should_refresh_cache(self)`
    - `clear_cache(self, user_id)` **Key Imports:** 8 modules

backend-api/app/services/privacy_scoring.py **Lines:** 297
Type: Python Module

Description: Enhanced Privacy Scoring Service Calculates comprehensive privacy score based on all privacy components...

Classes:

- `PrivacyScoring` (line 17)
  - Calculate and track comprehensive privacy scores
  - Methods: 3
    - `__init__(self)`
    - `_get_privacy_level(self, score)`
    - `_get_trend_message(self, trend)` **Key Imports:** 8 modules

backend-api/app/services/recommendation_engine.py **Lines:** 483
Type: Python Module

Description: Smart Recommendation Engine AI-powered personalized recommendations based on user behavior patterns...

Classes:

- `RecommendationEngine` (line 12)
  - Generates personalized recommendations using:
- Usage patterns analysis
- Time-based behavioral insi
  - Methods: 18

        * `__init__(self)`
        * `_analyze_patterns(self, analytics_data)`
        * `_identify_peak_hours(self, analytics_data)`
        * `_categorize_app_usage(self, analytics_data)`
        * `_analyze_focus_sessions(self, analytics_data)` **Key Imports:** 7 modules

backend-api/app/services/smart_replies.py **Lines:** 397
Type: Python Module

Description: Smart Reply Suggestions AI-powered quick response generation for notifications...

Classes:

- `ReplyType` (line 11)
  - Types of smart replies
- `SmartReplyGenerator` (line 20)
  - Generate contextual quick reply suggestions
  - Methods: 13
    * `__init__(self)`
    * `generate_replies(self, message, sender, app_name, context)`
    * `_detect_patterns(self, message)`
    * `_get_question_replies(self, message)`
    * `_get_meeting_replies(self)` **Key Imports:** 5 modules

backend-api/app/services/vpn_manager.py **Lines:** 271
Type: Python Module

Description: VPN Manager Service Handles VPN connection, monitoring, and leak detection...

Classes:

- `VPNStatus` (line 17)
- `VPNProtocol` (line 24)
- `VPNManager` (line 30)
  - Manage VPN connections and monitor for privacy leaks
  - Methods: 1
    * `__init__(self)` **Key Imports:** 9 modules

Core Modules

backend-api/app/core/cache.py **Lines:** 145
Type: Python Module

Description: Caching Configuration and Utilities Redis-based caching for improved API performance...

Classes:

- `CacheManager` (line 15)
  - Manages caching operations with Redis
  - Methods: 3
    * `__init__(self, redis_client)`
    * `_generate_key(self, prefix)`
    * `get_stats(self)` **Functions:**
- `cached(ttl, key_prefix)` (line 105)
  - Decorator for caching function results

Args: ttl: Time to live in seconds (default 5 minutes)

Key Imports: 6 modules

backend-api/app/core/database.py **Lines:** 102
Type: Python Module

Description: Database Configuration with Connection Pooling Optimized database settings for production performance...

Functions:

- `before_cursor_execute(conn, cursor, statement, parameters, context, executemany)` (line 50)
  – Log slow queries
- `after_cursor_execute(conn, cursor, statement, parameters, context, executemany)` (line 56)
  – Log query execution time
- `get_db()` (line 67)
  – Get database session with automatic cleanup Usage: db: Session = Depends(get_db)
- `check_database_connection()` (line 80)
  – Check if database connection is healthy
- `get_pool_stats()` (line 93)
  – Get connection pool statistics **Key Imports:** 9 modules

Mobile Application

React Native mobile application for Android with offline-first architecture.

Screens

mobile-app/src/screens/AnalyticsScreen.js **Lines:** 760
Type: JavaScript/React

React Components:

- `AnalyticsScreen` - Props: **React Hooks:**
- useState: `loading`
- useState: `refreshing`
- useState: `selectedTab`
- useState: `dashboardData`
- useState: `error`
- useEffect: 1 effects **Imports:** 4 modules

mobile-app/src/screens/FocusModeScreen.js **Lines:** 603
Type: JavaScript/React

React Components:

- `FocusModeScreen` - Props: **React Hooks:**
- useState: `hasPermission`
- useState: `isActive`
- useState: `currentSession`
- useState: `remainingMinutes`
- useState: `progress`
- useState: `stats`
- useState: `loading`
- useState: `selectedDuration`
- useEffect: 1 effects **Imports:** 2 modules

mobile-app/src/screens/GoalsScreen.js **Lines:** 534
Type: JavaScript/React

React Components:

- `GoalsScreen` - Props: **React Hooks:**
- useState: `goals`
- useState: `loading`
- useState: `modalVisible`
- useState: `newGoal`
- useEffect: 1 effects **Imports:** 3 modules

mobile-app/src/screens/HomeScreen.js **Lines:** 454
Type: JavaScript/React

React Components:

- `HomeScreen` - Props: { navigation } **React Hooks:**
- useState: `stats`
- useState: `privacyStatus`
- useState: `focusModeActive`
- useState: `refreshing`
- useState: `loading`
- useState: `sensorData`
- useState: `mqttConnected`
- useEffect: 1 effects **Imports:** 4 modules

mobile-app/src/screens/LoginScreen.js **Lines:** 211
Type: JavaScript/React

React Components:

- `LoginScreen` - Props: { navigation } **React Hooks:**
- useState: `email`
- useState: `password`
- useState: `isLoading` **Imports:** 2 modules

mobile-app/src/screens/NotificationsScreen.js **Lines:** 554
Type: JavaScript/React

React Components:

- `NotificationsScreen` - Props:
- `SwipeableNotificationCard` - Props: { item } **React Hooks:**
- useState: `notifications`
- useState: `filter`
- useState: `loading`
- useState: `refreshing`
- useState: `hasPermission`

- useEffect: 2 effects **Imports:** 3 modules

mobile-app/src/screens/PrivacyDashboardScreen.js **Lines:** 1029
Type: JavaScript/React

React Components:

- `PrivacyDashboardScreen` - Props: **React Hooks:**
- useState: `isVpnConnected`
- useState: `vpnStats`
- useState: `privacyScore`
- useState: `trackerStats`
- useState: `appPermissions`
- useState: `blockedDomains`
- useState: `whitelistedDomains`
- useState: `isLoading`
- useState: `refreshing`
- useState: `activeTab` **Imports:** 2 modules

mobile-app/src/screens/PrivacyScreen.js **Lines:** 377
Type: JavaScript/React

React Components:

- `PrivacyScreen` - Props: **React Hooks:**
- useState: `privacyStatus`
- useState: `loading`
- useEffect: 1 effects **Imports:** 2 modules

mobile-app/src/screens/RecommendationsScreen.js **Lines:** 810
Type: JavaScript/React

React Components:

- `RecommendationsScreen` - Props: { navigation } **React Hooks:**
- useState: `recommendations`
- useState: `filteredRecommendations`
- useState: `selectedCategory`
- useState: `selectedRecommendation`
- useState: `refreshing`
- useState: `loading`
- useState: `stats`
- useEffect: 2 effects **Imports:** 3 modules

mobile-app/src/screens/RegisterScreen.js **Lines:** 261
Type: JavaScript/React

React Components:

- `RegisterScreen` - Props: { navigation } **React Hooks:**
- useState: `fullName`
- useState: `email`
- useState: `password`
- useState: `confirmPassword`
- useState: `isLoading` **Imports:** 2 modules

mobile-app/src/screens/SettingsScreen.js **Lines:** 387
Type: JavaScript/React

React Components:

- `SettingsScreen` - Props: **React Hooks:**
- useState: `apiUrl`
- useState: `mqttBroker`
- useState: `mqttPort`
- useState: `notifications`
- useState: `darkMode`
- useState: `autoSync` **Imports:** 2 modules

Components

mobile-app/src/components/ErrorBoundary.js **Lines:** 148
Type: JavaScript/React

Imports: 1 modules

mobile-app/src/components/OfflineIndicator.js **Lines:** 45
Type: JavaScript/React

React Components:

- `OfflineIndicator` - Props: **Imports:** 3 modules

mobile-app/src/components/SkeletonLoader.js **Lines:** 172
Type: JavaScript/React

React Components:

- `SkeletonLoader` - Props: { width = '100%', height = 20, borderRadius = 4, style }
- `CardSkeleton` - Props:
- `StatCardSkeleton` - Props:
- `SensorCardSkeleton` - Props:
- `NotificationSkeleton` - Props:
- `DashboardSkeleton` - Props: **React Hooks:**
- useEffect: 1 effects **Imports:** 1 modules

Services

mobile-app/src/services/analytics.js **Lines:** 253
Type: JavaScript/React

Imports: 3 modules

mobile-app/src/services/api.js **Lines:** 482
Type: JavaScript/React

Imports: 5 modules

mobile-app/src/services/apiClient.js **Lines:** 262
Type: JavaScript/React

Imports: 2 modules

mobile-app/src/services/focusMode.js **Lines:** 431
Type: JavaScript/React

Imports: 2 modules

mobile-app/src/services/mqtt.js **Lines:** 202
Type: JavaScript/React

mobile-app/src/services/notifications.js **Lines:** 337
Type: JavaScript/React

Imports: 3 modules

mobile-app/src/services/privacy.js **Lines:** 619
Type: JavaScript/React

Imports: 2 modules

mobile-app/src/services/recommendations.js **Lines:** 456
Type: JavaScript/React

Imports: 2 modules

AI/ML Models

TensorFlow-based machine learning models for intelligent classification and prediction.

ai-models/init.py **Lines:** 0
Type: Python Module

ai-models/data/behavior_report.json **Lines:** N/A
Type: Python Module

ai-models/models/model_metadata.json **Lines:** N/A
Type: Python Module

ai-models/models/tflite_metadata.json **Lines:** N/A
Type: Python Module

ai-models/models/versions.json **Lines:** N/A
Type: Python Module

ai-models/training/init.py **Lines:** 0
Type: Python Module

ai-models/training/behavior_analyzer.py **Lines:** 446
Type: Python Module

Description: AI Model - User Behavior Analysis Tracks and analyzes user behavior patterns for personalization Provides insights into productivity, focus, and wellbeing trends...

Classes:

- `UserBehaviorAnalyzer` (line 14)
  - Analyze user behavior patterns and trends
  - Methods: 17
    - `__init__(self, data_path)`
    - `track_focus_session(self, start_time, end_time, quality_score)`
    - `track_distraction(self, timestamp, source, severity)`
    - `track_notification(self, timestamp, app_name, priority_score, was_handled)`
    - `track_app_usage(self, app_name, duration_minutes, timestamp)` **Functions:**
- `demo_behavior_analyzer()` (line 367)
  - Demo the behavior analyzer **Key Imports:** 7 modules

ai-models/training/context_suggestion_engine.py **Lines:** 435
Type: Python Module

Description: AI Model - Context-Aware Suggestion Engine Provides personalized suggestions based on user context and behavior Uses rule-based + ML hybrid approach for intelligent recommendations...

Classes:

- `ContextAwareSuggestionEngine` (line 13)
  - Generate context-aware suggestions for user wellbeing
  - Methods: 7
    - `__init__(self, model_path)`
    - `analyze_context(self, user_data)`
    - `generate_suggestions(self, user_data, max_suggestions)`
    - `_calculate_priority(self, category, confidence)`
    - `get_contextual_actions(self, suggestion)` **Functions:**
- `demo_suggestion_engine()` (line 328)
  - Demo the suggestion engine **Key Imports:** 6 modules

ai-models/training/convert_to_tflite.py **Lines:** 309
Type: Python Module

Description: TensorFlow Lite Model Converter Converts sklearn Random Forest model to TensorFlow Lite format for mobile deployment...

Classes:

- `TFLiteConverter` (line 14)
  - Convert sklearn model to TensorFlow Lite format
  - Methods: 8
    - `__init__(self, models_dir)`
    - `load_sklearn_model(self)`
    - `create_tensorflow_model(self)`
    - `train_tensorflow_model(self, model, num_samples)`
    - `convert_to_tflite(self, model, optimize)` **Functions:**
- `main()` (line 270)
  - Main conversion script **Key Imports:** 8 modules

ai-models/training/tflite_inference.py **Lines:** 343
Type: Python Module

Description: TFLite Mobile Inference Wrapper Provides inference interface for TensorFlow Lite models on mobile devices...

Classes:

- `TFLiteMobileInference` (line 14)
  - Mobile-optimized inference wrapper for TFLite models
  - Methods: 8
    - `__init__(self, models_dir)`
    - `load_model(self, model_name)`
    - `preprocess_text(self, text)`
    - `predict(self, text)`
    - `predict_batch(self, texts)`
- `TFLiteValidator` (line 210)
  - Validate TFLite model against sklearn model
  - Methods: 3
    - `__init__(self, models_dir)`
    - `load_models(self)`
    - `validate_predictions(self, test_texts)` **Functions:**
- `main()` (line 274)
  - Test TFLite mobile inference **Key Imports:** 10 modules

ai-models/training/train_focus_predictor.py **Lines:** 374
Type: Python Module

Description: AI Model Training - Focus Time Predictor Predicts optimal focus times based on user behavior patterns Learns when user is most productive and least distracted...

Classes:

- `FocusTimePredictor` (line 17)
  - Predict optimal focus periods for the user
  - Methods: 13
    - `__init__(self, model_path)`
    - `extract_temporal_features(self, hour, day_of_week)`
    - `extract_behavioral_features(self, avg_distractions, avg_screen_time, avg_notifications, recent_productivity)`
    - `_categorize_distractions(self, count)`
    - `_categorize_screen_time(self, minutes)` **Key Imports:** 13 modules

ai-models/training/train_notification_classifier.py **Lines:** 195
Type: Python Module

Description: AI Model Training - Notification Classifier Classifies notifications as urgent or non-urgent...

Classes:

- `NotificationClassifier` (line 15)
  - Train a model to classify notifications as urgent or non-urgent
  - Methods: 5
    - `__init__(self, model_path)`
    - `generate_training_data(self, num_samples)`
    - `train(self, df)`
    - `save_model(self)`
    - `predict(self, notification_text)` **Functions:**
- `main()` (line 163)
  - Main training script **Key Imports:** 8 modules

ai-models/training/train_priority_model.py **Lines:** 358
Type: Python Module

Description: AI Model Training - Notification Priority Scorer Scores notifications on a scale of 0-100 for priority Uses ML to learn user's notification importance patterns...

Classes:

- `NotificationPriorityScorer` (line 18)
  - Train a model to score notification priority (0-100)
  - Methods: 11
    - `__init__(self, model_path)`
    - `extract_temporal_features(self, timestamp)`
    - `extract_text_features(self, text)`
    - `extract_app_features(self, app_name)`
    - `generate_training_data(self, num_samples)` **Key Imports:** 11 modules

IoT Device

Raspberry Pi-based IoT device with environmental sensors.

iot-device/mqtt_client.py **Lines:** 174
Type: Python Module

Description: IoT Device - MQTT Client Raspberry Pi sensor monitoring and automation...

Classes:

- `WellbeingIoTDevice` (line 25)
  - Main IoT device class for environmental monitoring
  - Methods: 10
    - `__init__(self)`
    - `on_connect(self, client, userdata, flags, rc)`
    - `on_message(self, client, userdata, msg)`
    - `handle_command(self, command)`
    - `read_sensors(self)` **Key Imports:** 8 modules

iot-device/sensors/init.py **Lines:** 1
Type: Python Module

Description: Sensors package...

iot-device/sensors/dht_sensor.py **Lines:** 82
Type: Python Module

Description: DHT22 Temperature and Humidity Sensor Module...

Classes:

- `DHTSensor` (line 9)
  - DHT22 temperature and humidity sensor interface
  - Methods: 4
    - `__init__(self, gpio_pin)`
    - `read_temperature(self)`
    - `read_humidity(self)`
    - `cleanup(self)` **Key Imports:** 5 modules

iot-device/sensors/light_sensor.py **Lines:** 50
Type: Python Module

Description: TSL2561 Light Sensor Module Measures ambient light level in lux...

Classes:

- `LightSensor` (line 10)
  - TSL2561 light sensor interface
  - Methods: 3
    - `__init__(self, i2c_address)`
    - `read_lux(self)`
    - `cleanup(self)` **Key Imports:** 4 modules

iot-device/sensors/noise_sensor.py **Lines:** 85
Type: Python Module

Description: Noise Sensor Module Measures ambient sound level using USB microphone...

Classes:

- `NoiseSensor` (line 11)
  - USB microphone noise level sensor
  - Methods: 3
    - `__init__(self, device_index)`
    - `read_db(self)`
    - `cleanup(self)` **Key Imports:** 4 modules

iot-device/sensors/pir_sensor.py **Lines:** 50
Type: Python Module

Description: PIR Motion Sensor Module Detects motion using HC-SR501 PIR sensor...

Classes:

- `PIRSensor` (line 10)
  - PIR motion sensor interface
  - Methods: 3
    - `__init__(self, gpio_pin)`
    - `read(self)`
    - `cleanup(self)` **Key Imports:** 3 modules

iot-device/sensors/sensor_manager.py **Lines:** 150
Type: Python Module

Description: Sensor Manager Aggregates all sensor readings into a single interface...

Classes:

- `SensorManager` (line 15)
  - Manages all sensors and aggregates readings
  - Methods: 6
    - `__init__(self)`
    - `read_all(self)`
    - `_get_default_readings(self)`
    - `analyze_environment(self)`
    - `_calculate_environment_score(self, data)` **Key Imports:** 6 modules

Tests

Comprehensive test suite covering backend, mobile, and AI components.

Total Test Files: 26

Configuration Files

models/model_metadata.json

Configuration Keys: model_type, features, classes, version

scripts/generate_documentation.py