deploymentDiagram  
 title Pro Mood Tracker - Deployment Architecture  
  
 node "User Devices" {  
 component[Android Device] as androidDevice  
 component[iOS Device] as iosDevice  
 component[Flutter Application] as flutterApp  
 }  
  
 node "Firebase Cloud Platform" {  
 node "Authentication Services" {  
 component[Firebase Auth] as firebaseAuth  
 }  
   
 node "Database Services" {  
 component[Cloud Firestore] as firestore  
 component[Realtime Database] as realtimeDB  
 }  
   
 node "Storage Services" {  
 component[Cloud Storage] as cloudStorage  
 }  
   
 node "Analytics & Monitoring" {  
 component[Firebase Analytics] as firebaseAnalytics  
 component[Crashlytics] as crashlytics  
 component[Performance Monitoring] as perfMonitoring  
 }  
   
 node "Serverless Functions" {  
 component[Cloud Functions] as cloudFunctions  
 }  
 }  
   
 node "External API Services" {  
 component[Weather API] as weatherAPI  
 component[Location Services] as locationServices  
 component[Push Notification Service] as pushNotification  
 }  
   
 androidDevice -- flutterApp  
 iosDevice -- flutterApp  
   
 flutterApp --> firebaseAuth : "Authentication"  
 flutterApp --> firestore : "Data Storage & Sync"  
 flutterApp --> realtimeDB : "Real-time Updates"  
 flutterApp --> cloudStorage : "Media Storage"  
 flutterApp --> firebaseAnalytics : "Usage Metrics"  
 flutterApp --> crashlytics : "Crash Reporting"  
 flutterApp --> weatherAPI : "Weather Data"  
 flutterApp --> locationServices : "Location Data"  
   
 cloudFunctions --> firestore : "Trigger on Data Change"  
 cloudFunctions --> pushNotification : "Send Notifications"  
 cloudFunctions --> weatherAPI : "Scheduled Data Fetch"

## Figure 4.12: Deployment Diagram - Pro Mood Tracker Application

This deployment diagram illustrates the infrastructure and services required to deploy the Pro Mood Tracker application, showing how different components interact across various environments.

### Deployment Components:

1. **User Devices**:
   * Android and iOS mobile devices running the Flutter application
   * Native components integrated via Flutter plugins
   * Local SQLite database for offline storage
   * SharedPreferences for user settings
2. **Firebase Cloud Platform**:
   * Authentication Services:
     + Firebase Authentication for secure user identification
     + Social login providers integration (Google, Apple)
   * Database Services:
     + Cloud Firestore for structured data storage
     + Realtime Database for synchronization and real-time updates
   * Storage Services:
     + Cloud Storage for media files and backups
   * Analytics & Monitoring:
     + Firebase Analytics for user behavior tracking
     + Crashlytics for error reporting and diagnostics
     + Performance Monitoring for application efficiency metrics
   * Serverless Functions:
     + Cloud Functions for backend processing
     + Scheduled tasks for data aggregation
     + Event-triggered functions for notifications
3. **External API Services**:
   * Weather API for environmental context data
   * Location Services for geographical information
   * Push Notification Service for user engagement

### Deployment Characteristics:

1. **Client-Side Deployment**:
   * Cross-platform deployment via Flutter
   * Compiled to native code for each target platform
   * Optimized for performance on mobile devices
   * Responsive design for various screen sizes
2. **Server-Side Architecture**:
   * Serverless architecture using Firebase services
   * Event-driven processing using Cloud Functions
   * Scalable storage with automatic sharding
   * Secure data access with authentication and authorization rules
3. **Data Flow**:
   * Bidirectional synchronization between devices and cloud
   * Offline-first approach with local-first data storage
   * Periodic syncing when connectivity is available
   * Real-time updates for critical information
4. **Security Measures**:
   * End-to-end encryption for sensitive data
   * Token-based authentication for API access
   * Data validation at multiple levels
   * Rule-based access control in Firestore
5. **Scalability Provisions**:
   * Auto-scaling of Firebase services
   * Load balancing for high traffic
   * Database indexing for query optimization
   * Caching mechanisms for frequently accessed data

This deployment architecture enables the Pro Mood Tracker application to function reliably across different devices while providing a seamless user experience with both online and offline capabilities. The serverless backend minimizes operational overhead while providing robust scalability to accommodate growing user numbers.