**Final System Architecture: Campus Cycle**

This is a comprehensive and detailed system architecture for the Campus Cycle project, integrating all features from your provided documents and designed around the core requirements of using **Android Studio** and **Firebase**.

**1. Architectural Model Overview**

The system operates on a **client-server model** with a **serverless backend**, as illustrated in the provided diagram. The Android application acts as the client, while Firebase services provide the backend infrastructure, a design that ensures high scalability, reliability, and minimal operational overhead.

**2. Layered Component Breakdown & Tech Stack**

The architecture is divided into three primary layers: the Mobile Application (Client), the Firebase Services (Backend), and the Data Storage Layer.

**A. Mobile Application (Client-Side) 📱**

The client is the **Android application** developed in **Android Studio** using the **Java** programming language.

* **User Interface (UI) & Presentation Layer**:
  + **Purpose**: Manages the user's interaction with the app.
  + **Tech Stack**: **Android XML layouts** for UI design and **Java** for activity logic.
  + **Relevant Java Files**:
    - MainActivity.java: The primary entry point for the app, handling user authentication and navigation to the main dashboard.
    - DashboardActivity.java: Displays the central hub with options for reporting, tracking status, and viewing the personal waste footprint.
    - ComplaintActivity.java: The form for submitting a complaint, handling image capture, location data, and form validation.
    - StatusAdapter.java: A custom adapter for a RecyclerView that displays the user's submitted complaints and their real-time statuses.
    - MyProfileActivity.java: Displays the Personal Waste Footprint Dashboard data.
* **Business Logic Layer**:
  + **Purpose**: Executes the core functionality of the application on the device.
  + **Tech Stack**: **Java** with the **TensorFlow Lite Android library**.
  + **Relevant Java Files**:
    - WasteClassifier.java: A utility class that loads and runs the pre-trained .tflite machine learning model. It processes an image taken by the user and returns the classified waste type.
    - LocationService.java: Uses Android's native location APIs (FusedLocationProviderClient) to get the user's GPS coordinates for complaint reports and Proximity-Based Smart Bin Notifications.
    - NotificationService.java: A custom service that listens for and handles incoming push notifications from Firebase Cloud Messaging (FCM).
* **Firebase Integration Layer**:
  + **Purpose**: Handles all communication between the app and the Firebase backend services.
  + **Tech Stack**: **Firebase Android SDK**.
  + **Relevant Java Files**:
    - FirebaseManager.java: A central class that handles authentication, database writes, and file uploads. It abstracts the complexity of interacting with different Firebase services.

**B. Firebase Services (Backend) ☁️**

This layer is a set of managed, serverless services that provide the backend functionality, replacing the need for a custom server.

* **Authentication Service**: **Firebase Authentication**.
  + **Purpose**: Manages user registration, login, and session management. It provides secure and scalable user management out of the box.
* **Complaint & Routing Service**: **Firebase Cloud Functions**.
  + **Purpose**: Contains the server-side business logic for complaint processing and route optimization.
  + **Implementation**: These are event-driven functions triggered by actions in the app. They are written in **Java**.
  + **Relevant Cloud Functions**:
    - NewComplaintTrigger.java: Triggered when a new document is added to the complaints collection in Firestore. It identifies the complaint type (campus or hostel), finds the appropriate cleaner or warden, and sends a push notification to their device via FCM.
    - RouteOptimizerFunction.java: A callable function that a staff member can trigger. It retrieves all unassigned complaints from Firestore, runs a custom route optimization algorithm, and returns the most efficient route.
* **Predictive Analytics Service**: **Firebase Cloud Functions**.
  + **Purpose**: Manages the Predictive Litter Hotspot Mapping feature.
  + **Implementation**: A scheduled function that periodically runs a predictive algorithm on historical data from Firestore.
  + **Relevant Cloud Function**:
    - HotspotPredictor.java: This function analyzes historical complaint data and saves the predicted hotspot locations back to a separate collection in Firestore for display on the admin and staff dashboards.
* **Notification Service**: **Firebase Cloud Messaging (FCM)**.
  + **Purpose**: Manages and sends real-time push notifications to user devices. It is integrated with the Cloud Functions to notify users when a complaint status changes or when a new complaint is assigned to a staff member.

**C. Data Storage Layer 💾**

* **Structured Data**: **Cloud Firestore**.
  + **Purpose**: Stores all structured data in a scalable NoSQL database.
  + **Collections**:
    - users: Stores user profiles.
    - complaints: Stores details of each complaint, including type, status, and location.
    - bins: Stores the location and current fill level of smart bins.
    - hotspots: A separate collection to store the data generated by the HotspotPredictor function.
* **Unstructured Data**: **Cloud Storage for Firebase**.
  + **Purpose**: A highly scalable object storage service for all user-uploaded photos and videos of waste.