Component	Framework/Tool	Purpose	Role in the App
UI Development	BeeWare (Toga)	Build cross-platform, native-looking user interfaces.	Create the app's layout, buttons, menus, input fields, and other user interactions using native widgets.
Remote Storage	PostgreSQL	Centralized, server-based relational database for handling larger datasets and multiple users.	Manage shared data (e.g., user accounts, classification results) on a backend server.
Backend Development	Flask (or Django)	Build backend APIs to manage communication between the mobile app and server.	Provide RESTful APIs for uploading images, syncing data, managing user authentication, and fetching classification results.
Basic Image Processing	Pillow	Perform simple image processing tasks like resizing, cropping, and format conversion.	Pre-process images before storing them in the database or sending them for further analysis.
Advanced Image Processing	OpenCV	Handle complex image and video processing tasks such as motion detection and filtering.	Analyze video frames and images for preprocessing and motion detection before machine learning classification.
Object Detection & ML	YOLO (You Only Look Once)	Perform real-time object detection and classification on images or video frames.	Identify animals in images or videos captured by the app, and classify objects in real time.
Real-Time Notifications	Firebase Cloud Messaging (FCM)	Send real-time push notifications to users.	Notify users when an animal is detected or data is synced with the server.
Bluetooth Communication	PyBluez	Enable Bluetooth communication with external devices.	Connect the mobile app to trail cameras or sensors via Bluetooth, especially when there’s no network connection.
WiFi Communication	PyWiFi	Handle wireless communication over WiFi.	Connect to local devices (e.g., trail cameras) via WiFi when Bluetooth is not available.
Network Communication	Requests / WebSockets	Manage HTTP requests and real-time communication between the mobile app and backend.	Send and receive data (e.g., images, classification results) between the mobile app and the backend server.
Offline/Online Syncing	Hybrid Use of SQLite and PostgreSQL	Sync data between local (SQLite) and remote (PostgreSQL) storage when the app regains network access.	Enable offline functionality using SQLite and synchronize data with a centralized PostgreSQL server when the device is connected to the internet.

Push Notifications	Firebase Cloud Messaging	Provide push notifications for real-time alerts.	Send alerts to users when animal detection events or classification results are triggered.
Data Analytics and Reporting	Pandas / Matplotlib	Process and analyze collected data, generate visual reports.	Analyze data, such as animal detection trends, and generate charts or graphs for user reporting.

Table Name	**Purpose**
User	Store user information and credentials.
User_Settings	Store personalized app settings for each user.
Camera	Store information about each camera associated with the system.
Camera_Node_Configuration	Store configuration settings for each camera node.
Classification_Test_Dataset	Store information on test datasets for machine learning models.
Training_Dataset	Store information on training datasets for machine learning models.
Previous_Videos	Store historical videos/images for user review.
Live_Feed_Metadata	Store metadata on live camera feeds.
Classification	Store the classification results of detected animals.
Detected_Animal_Logs	Log each detected animal for historical tracking.
Alerts	Store information about alerts generated for users.
Event_Log	Track user activities and app events.
Camera_Activity_Log	Track camera-related activities, such as recording and detections.
Storage_Sync_Status	Monitor the synchronization status between local and remote storage.

Table Name	**Purpose**	**Key Columns**
User	Store user information and credentials.	user_id (PK), username, email, password_hash, phone_number, app_configuration, created_at, last_login
User_Settings	Store personalized app settings for each user.	user_id (FK), alert_preferences, notification_settings, camera_access_list, motion_sensitivity
Camera	Store information about each camera associated with the system.	camera_id (PK), user_id (FK), camera_name, location, is_active, date_added

Camera_Node_Configuration	Store configuration settings for each camera node.	camera_id (FK), detection_zone, motion_sensitivity, animal_detection_enabled, recording_schedule	
Classification_Test_Dataset	Store information on test datasets for machine learning models.	test_id (PK), dataset_name, dataset_description, date_created, used_in_model_version	
Training_Dataset	Store information on training datasets for machine learning models.	training_id (PK), dataset_name, dataset_description, date_created, used_in_model_version	
Previous_Videos	Store historical videos/images for user review.	event_id (PK), camera_id (FK), video_path, image_path, event_type, date_recorded, duration, storage_location	
Live_Feed_Metadata	Store metadata on live camera feeds.	feed_id (PK), camera_id (FK), start_time, end_time, average_quality, is_recorded	
Classification	Store the classification results of detected animals.	classification_id (PK), event_id (FK), animal_detected, confidence_score, classification_time, classification_model_version	
Detected_Animal_Logs	Log each detected animal for historical tracking.	log_id (PK), camera_id (FK), animal_detected, detection_time, image_path, location, user_notified	
Alerts	Store information about alerts generated for users.	alert_id (PK), user_id (FK), camera_id (FK), alert_type, alert_message, timestamp, is_viewed, is_dismissed	
Event_Log	Track user activities and app events.	event_log_id (PK), user_id (FK), event_type, event_details, timestamp	
Camera_Activity_Log	Track camera-related activities, such as recording and detections.	activity_log_id (PK), camera_id (FK), activity_type, timestamp, details	
Storage_Sync_Status	Monitor the synchronization status between local and remote storage.	sync_id (PK), camera_id (FK), data_type, last_sync_time, sync_status, conflict_resolution	

Module Name	**Description**	**Purpose**	
User Interface (UI)	Develop the UI using BeeWare (Toga) to provide a native-looking app experience.	Allows users to interact with the app and manage cameras, view events, configure settings.	
Database Integration	Use PostgreSQL as the central database for all application needs.	Store user data, classification results, training and testing datasets, and historical video data. Ensure data integrity and scalability in a multi-user environment.	
Backend Development	Use Flask (or Django) to create RESTful APIs for app functionality.	Handle user authentication, data access, and classification requests.	
Image & Video Processing	Integrate Pillow, OpenCV, and YOLO for image preprocessing, video handling, and object detection.	Detect and classify animals in video feeds and trigger alerts based on detections.	
Camera & Device Integration	Set up Bluetooth/WiFi integration using PyBluez or PyWiFi for camera communication.	Enable communication with trail cameras even without network access.	

Alerts and Notifications	Use Firebase Cloud Messaging (FCM) to implement real-time user notifications.	Notify users when animals are detected in video feeds by the model.	
Training & Testing Dataset Management	Implement tables for managing training, testing datasets, and results.	Store labeled datasets for model training and evaluate model accuracy with test data.	
Classification Results	Implement separate tables for training/testing results and real-world deployed app results.	Differentiate between experimental model evaluations and live, deployed results.	