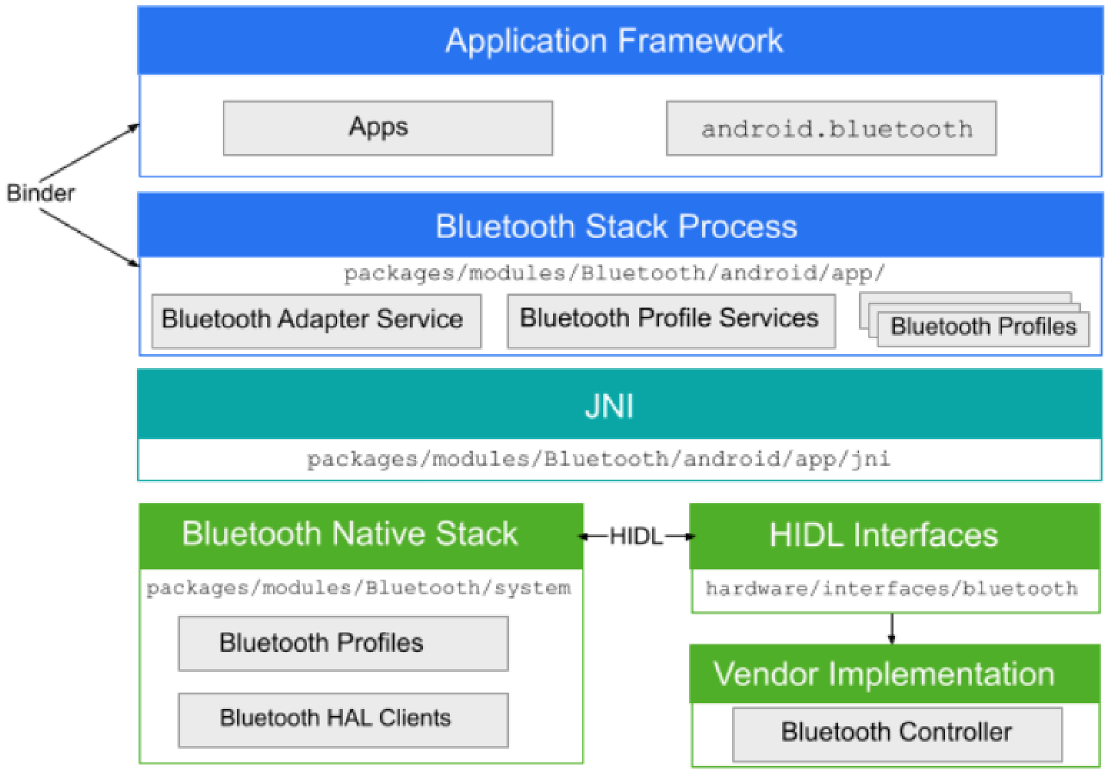
CPE495
G10: Alyson & Bobby

Bluetooth Protocols

- Android can support two Bluetooth Protocols:
    - Classic Bluetooth
        - Higher power consumption
        - Transmits data in a continuous stream
    - Bluetooth Low Energy
        - Less power consumption
        - Designed to transmit data in short bursts rather than continuously
        - Must have a BLE qualified chipset as it is not backwards compatible with older bluetooth chipsets
        - Available in Android 4.3 and later

Structure of Bluetooth Stack with Android Framework



Bluetooth : Android Open Source Project. Android Open Source Project. (2024, August 8). https://source.android.com/docs/core/connect/bluetooth

Similar Application: Ring App

A similar application that we can take inspiration from is the **Ring App**, known for its features related to home security and camera monitoring. The aspects of the Ring app that are particularly relevant to our project are its ability to manage and monitor multiple cameras and its user-friendly approach to camera control and settings. Key features of the Ring app that we would like to replicate include:

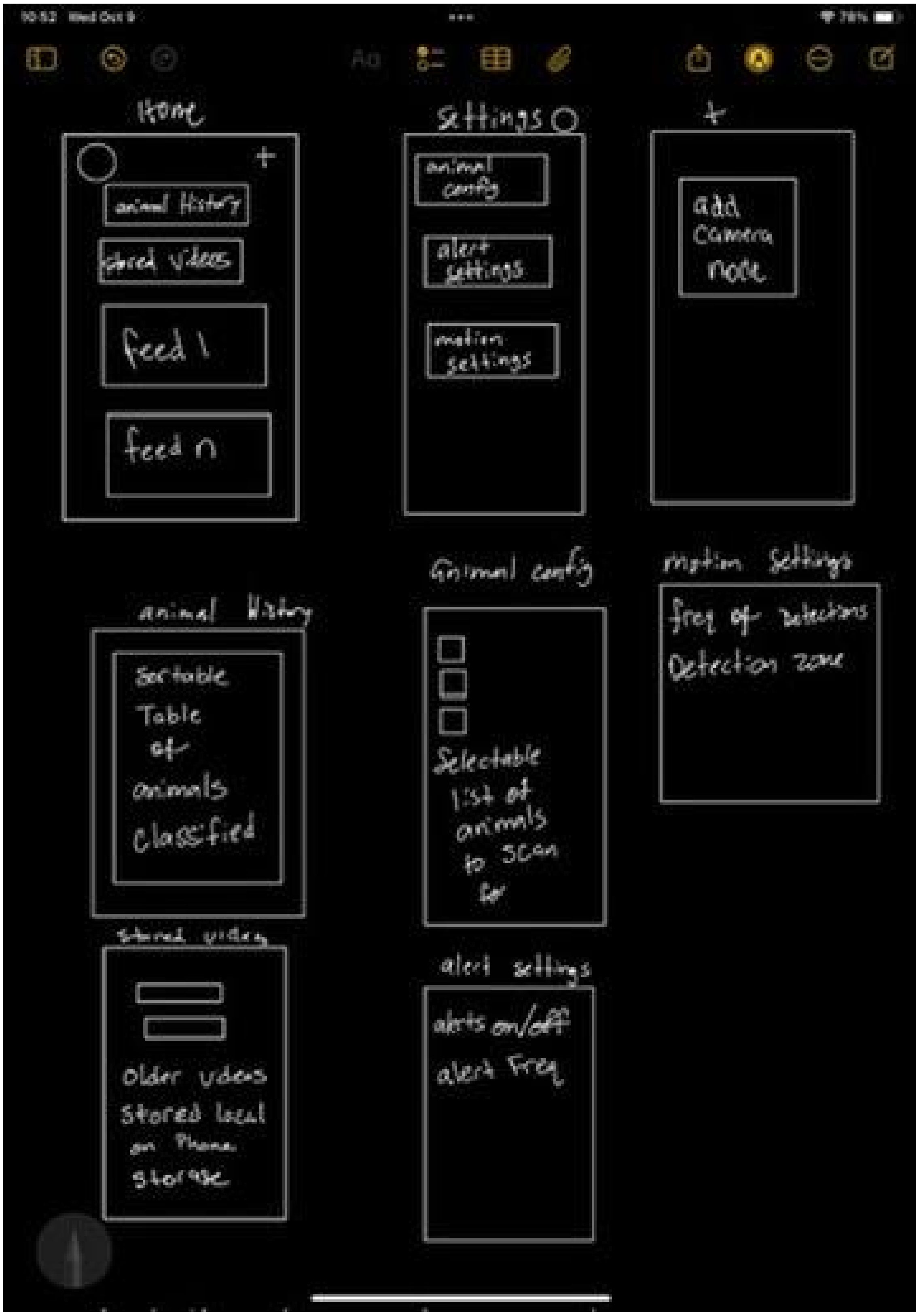
1. **Monitoring Multiple Cameras**:
    - The Ring app allows users to monitor **multiple camera feeds**, either **statically** (reviewing captured footage) or through **live feeds** in real-time. This is essential for monitoring different locations or zones simultaneously.
2. **Adding Cameras**:
    - Users can easily add and configure new cameras in the app. This ability to **add cameras** and seamlessly integrate them into the app for monitoring and control is crucial for our project, especially as users might have multiple cameras in different locations.
3. **Storing Captured Videos**:
    - The app provides cloud storage for **captured videos**, allowing users to store, review, and manage footage. This feature aligns with our need to store **captured images and videos** for later review or analysis, either locally or in the cloud.
4. **Viewing Previous Events**:
    - Ring enables users to view **previously captured events**, such as motion detection recordings or alerts. Our app should offer similar functionality, allowing users to easily access a **timeline of captured events** and view relevant images or videos.
5. **Robust Camera Settings**:
    - The Ring app includes detailed settings for controlling how each camera operates, such as **motion sensitivity, recording schedules, and alert preferences**. We aim to incorporate similarly **robust settings**, enabling users to fine-tune how their cameras function based on personal preferences or environmental conditions.

High Level Design/Framework Choices

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.
Local Storage	SQLite	Lightweight, embedded database for local data storage.	Store user preferences, locally cached data, and offline data on the mobile device.
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.

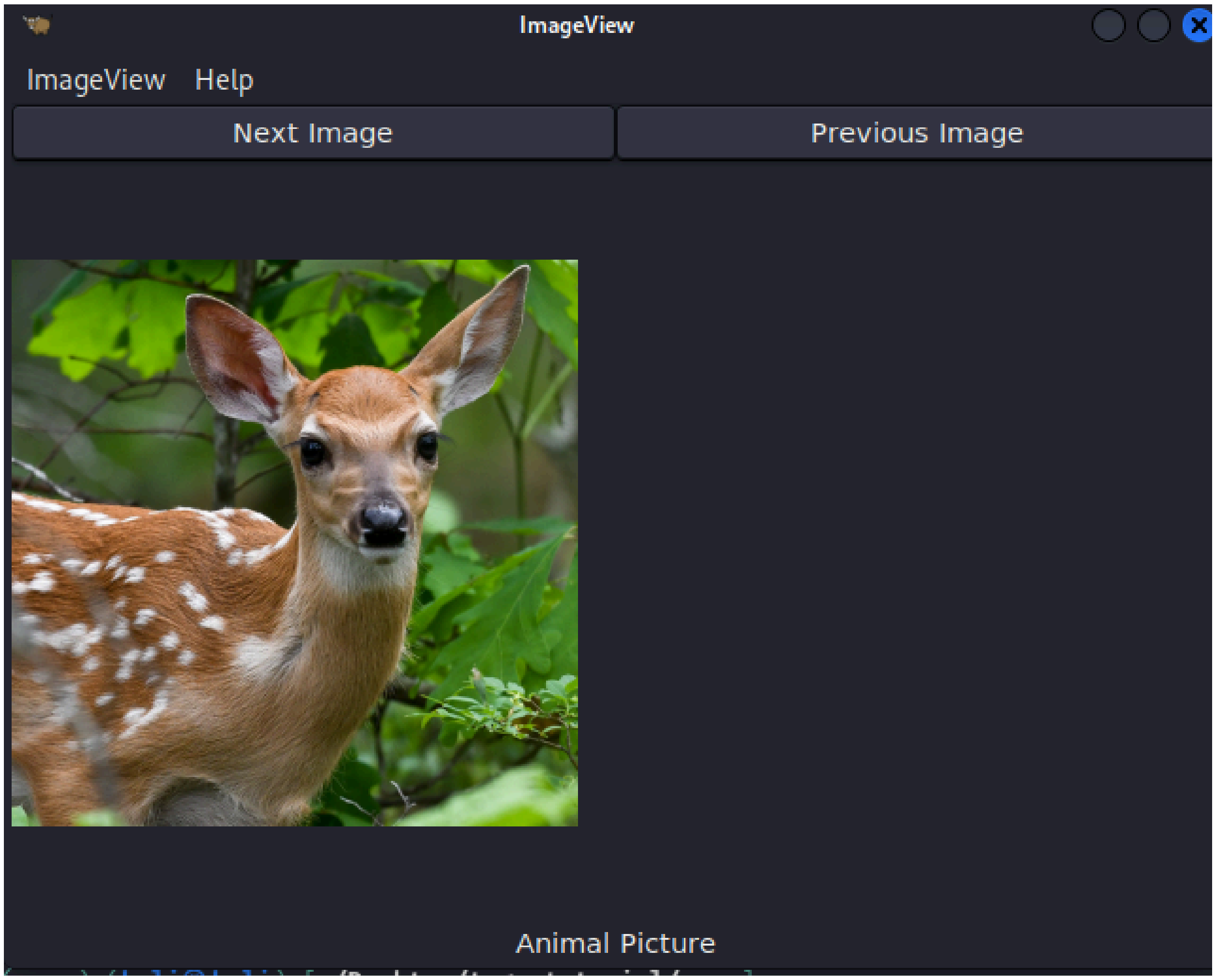
Basic UI Design



Overview of the UI Design Elements

1. **Home Screen**
   - **Animal History**: A button that directs users to a **sortable table of animals classified**. This feature is valuable for viewing and organizing all detected animals and understanding patterns.
   - **Stored Videos**: A section for viewing **locally stored videos**. This will let users access past footage and review previously detected events.
   - **Feed n (Dynamic Feeds)**: Buttons to view **live feeds** from multiple cameras. Allowing users to monitor multiple feeds directly is a key feature, similar to the Ring app, and this works well for your goal of providing simultaneous monitoring capabilities.
2. **Settings Menu**
   - **Animal Config**: A section where users can **select which animals** they are interested in monitoring. This feature allows users to customize the app's detection priorities, improving usability by tailoring the app to specific needs (e.g., only scanning for certain types of wildlife).
   - **Alert Settings**: Provides the ability to **turn alerts on or off** and adjust the **frequency of alerts**. This is essential for giving users control over how often they receive notifications, preventing alert fatigue.
   - **Motion Settings**: Allows users to configure the **frequency of detections** and set a **detection zone**. This gives more granular control over the camera's motion sensitivity and area coverage, which is useful for reducing false positives.
3. **Add Camera Node Screen**
   - **Add Camera Node**: A button to add new cameras, expanding the system. This feature is straightforward and vital for ensuring scalability so that users can add additional cameras as needed.

Screenshot of basic UI coded in toga.



High Level Database Table View

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.