# Development Diary

**15/2: Initial Web Dashboard Setup**

* Set up basic Streamlit project structure.
* Developed a mockup for the nutrition tracking dashboard with placeholder images and chat history.
* Challenges:
  + Establishing state management for chat history and file uploads.
  + Ensuring responsiveness on different screen sizes.

**17/2: Integrating User Authentication & API Key Generation**

* Implemented user registration and login using Flask-Login.
* Generated unique API keys for each user to secure communications from edge devices.
* Challenges:
  + Syncing session state across Streamlit and Flask.
  + Managing API key persistence.

**28/2: Setting Up Webcam Streaming Server**

* **Installed Flask and OpenCV** on the Raspberry Pi.
* **Connected USB webcam** and wrote a Python script to capture video frames.
* **Set up Flask server** to stream the video feed locally (no cloud dependency).
* **Challenges:**
  + Video streaming was choppy at first.
  + Adjusted webcam resolution and frame rate to improve performance.

**28/2: Testing Webcam Streaming**

* **Tested webcam stream** for stability and performance.
  + Video feed was still lagging, so further adjustments were made.
  + Improved frame rate and reduced resolution for smoother streaming.
* **Challenges:**
  + Performance was poor initially, requiring several tweaks to achieve a consistent stream.

**02/3: Model Training and Fine-Tuning (MiniCPM)**

* Started training the CNN-based food classification model using PyTorch.
* Fine-tuned the model on local hawker dishes to improve recognition accuracy.
* **Challenges:**
  + Balancing model size and inference speed for edge deployment.
  + Handling data variability in local food images.

**10/3: Load Cell with HX711 Setup**

* **Wired HX711 load cell amplifier** to Raspberry Pi GPIO pins.
* **Used HX711 Python library** to read data from the load cell.
* **Calibrated the load cell** with known weights for accurate readings.
* **Challenges:**
  + Load cell readings were noisy.
  + Implemented software filters to smooth data (average filtering).
  + Calibration drifted over time, requiring frequent adjustments.

**10/3: Testing Load Cell**

* **Tested load cell with various weights** to check for consistent readings.
* **Adjusted calibration** multiple times to ensure accurate weight measurement.
* **Challenges:**
  + Load cell required multiple recalibrations to minimize drift.
  + Filtering methods were fine-tuned to remove noise and stabilize readings.

**11/3: Integrating Webcam, Load Cell, Button, and LED**

* **Combined components**: webcam, load cell, button, and LED into a unified system.
* **Created Python script** to:
  + Start webcam streaming when button is pressed.
  + Turn on LED when weight threshold is exceeded.
* **Processed all data locally** on the Raspberry Pi.
* **Challenges:**
  + Managed limited GPIO pins for multiple components.
  + Ensured system remained responsive when button was pressed (avoiding blocking operations for video feed or weight measurement).

**11/3 - 28/3: Testing the Integrated System**

* **Tested integrated system for extended periods:**
  + Webcam stream remained stable after integration.
  + Weight readings were accurate, and LED responded correctly to weight threshold.
  + Button triggered both webcam stream and LED light as expected.
* **Full Pipeline Testing:**
  + Verified that the Pi could capture both image and weight data.
  + Tested the secure transmission of this data from the Pi to the webserver using the API key.
  + Confirmed that the backend correctly classified the image, calculated the nutritional values, and stored the record in the SQLite database.
* **Challenges:**
  + Coordinating end-to-end data flow to ensure reliable and timely communication between the edge device and the webserver.
  + Ensured no conflict between the button press triggering multiple actions (video streaming and weight checking).
  + Handled GPIO resource conflicts and managed processing times for simultaneous tasks.

**28/3: Final Testing and Optimization**

* **Final system test:**
  + **Webcam stream** stable with improved performance after fine-tuning.
  + **Weight readings** accurate with smooth data output.
  + **LED light** responded correctly when weight threshold was exceeded.
* **Optimized code** for cleaner execution and added basic error handling.
* **Challenges:**
  + Raspberry Pi began heating up after extended testing. Cooling solutions considered for future use.
  + Ngrok tunnel occasionally disconnected, but system resumed after restarting.