**Project Overview**

In this project, we developed a client-server application using Flask for image classification with a local deep learning model. The server-side application allows users to upload images from a web interface, which are then classified using the YOLOv8n model for object detection. Below is a breakdown of the key components, functionalities, choices made, and tests implemented.

**Code Files and Functionality**

1. **app.py**: Core server-side application handling user requests and image classification.
   * **Routes**:
     + **Login and Authentication**: Users must authenticate to access image upload functionalities.
     + **Image Upload**: Supports uploading images from devices or capturing them via mobile browsers.
     + **Classification**: Utilizes the YOLOv8n model (yolov8n.pt) to perform inference on uploaded images, extracting object detections.
     + **Response Handling**: Returns classification results in JSON format or renders them on a results page based on request headers.
2. **index.html**: Frontend interface utilizing JavaScript for image capture, upload functionality, and response display.
   * **User Experience**: Redirects unauthenticated users to the login page to ensure secure access to image upload features.
3. **login.html**: Login page where users enter credentials for authentication before accessing image upload functionalities.
4. **test\_app.py**: Unit tests validating server functionalities and ensuring application robustness.
   * **Test Cases**:
     + **test\_login\_page**: Verifies the successful loading of the login page.
     + **test\_upload\_image**: Tests image upload functionality, ensuring correct handling and classification of images.
     + **test\_status**: Checks the status endpoint to confirm server health and response integrity.
5. **requirements.txt**: Lists project dependencies including Flask, Torch, and PIL necessary for running the application.

**Choices Made**

* **YOLOv8n Model**: Selected the YOLOv8n model for object detection due to its efficiency and accuracy in real-time object detection tasks.
  + **Implementation**: Loaded the YOLOv8n model (yolov8n.pt) using the YOLO class to perform inference on uploaded images.
  + **Object Detection**: Extracted object bounding boxes and class probabilities (detections) to identify and classify objects within images.

**Tests Created**

* **test\_login\_page**:
  + **Purpose**: Ensures the login page loads successfully for user authentication.
  + **Validation**: Checks if the response status code is 200 and verifies the presence of 'Login' in the response data.
* **test\_upload\_image**:
  + **Purpose**: Validates image upload functionality, confirming proper handling and classification of uploaded images.
  + **Assertion**: Verifies that the server responds with a status code of 200 upon successful image upload.
* **test\_status**:
  + **Purpose**: Tests the status endpoint to ensure server health and functionality.
  + **Validation**: Confirms the status endpoint returns a status code of 200 and includes 'status' in the response data to indicate operational status.