Knight Lab Intern Web Application

Yehia Hany

June 30, 2025

1. Application Structure and Workflow

The application follows a modular and scalable structure, built using the Flask framework with a clear separation between the frontend, backend, and AI model layers. Below is a summary of the application's workflow:

- **User Authentication**: Users access the system via a combined login/signup interface. Credentials are securely validated on the backend.
- **Upload and Analysis**: After logging in, users can upload an image to trigger the AI pipeline.
- **Model Execution**: The server processes the image using deep learning models for classification, object detection, and semantic segmentation.
- **Results Display**: The processed images and results are rendered dynamically using HTML templates with embedded base64 images.

The application uses Flask blueprints for routing, SQLAlchemy for database interactions, and Flask-Login for session management. The entire workflow is secured for deployment on HTTPS platforms like Hugging Face Spaces.

2. Technologies and Tools Used

Backend

- **Flask** – Web framework for Python
- **Flask-Login** – User session management
- **Flask-SQLAlchemy** – ORM for database access
- **SQLite** – Lightweight database

Frontend

- **HTML5, CSS3, JavaScript** – Core technologies
- **Bootstrap (Nova Template)** – Responsive UI design
- **SweetAlert2** – Modern alert and toast notifications

AI and ML Models

- **PyTorch** – Deep learning framework
- **ResNet-50** – Image classification (ImageNet)
- **YOLOv5s (Ultralytics)** – Object detection
- **DeepLabV3** – Semantic segmentation
- **Torchvision & PIL** – Image preprocessing and visualization

3. Model Integration Process

The core ML pipeline is implemented in a utility module (`utils/predict.py`). The process includes:

1. **Image Classification**: The uploaded image is resized and passed to a pretrained ResNet-50 model to predict the most likely object label from the ImageNet dataset.
2. **Object Detection**: YOLOv5s is used to detect all objects in the image and draw bounding boxes with confidence scores.
3. **Semantic Segmentation**: DeepLabV3-ResNet50 segments the image and retains only animal-related pixels (e.g., dogs, cats, birds) to produce a binary mask.
4. **Result Packaging**: All three images (classification, detection, segmentation) are converted to base64 and embedded into the output HTML.

4. Summary of Main Features

- **Combined Login/Signup Interface**: A single page handles both actions using JavaScript toggling and AJAX requests.
- **SweetAlert2 Integration**: Clean, non-blocking feedback system for success, errors, and notices.
- **Mobile-Responsive Design**: Based on a modified Nova template with improved usability on small screens.
- **Model Pipeline Integration**: Real-time classification, detection, and segmentation using PyTorch models.
- **Secure Session Handling**: Encrypted cookies, HTTPS-only flags, and server-side authentication with Flask-Login.
- **Deployment Ready**: Supports hosting on Hugging Face Spaces or any HTTPS-compatible platform.

Authentication Security

The application implements a secure user authentication system using Flask-Login and Werkzeug's security module. During the signup process, user passwords are not stored in plain text. Instead, they are securely hashed using the `pbkdf2:sha256` algorithm provided by `werkzeug.security`. This ensures that even if the database is compromised, user passwords remain protected.

During login, the password entered by the user is verified using the `check_password_hash()` function, which compares the hash of the entered password with the stored hash. Email inputs are normalized (trimmed and lowercased) to prevent duplication and ensure consistency.

Additional validation includes:

- Ensuring email format is valid using regular expressions.
- Enforcing minimum length requirements for email, name, and password fields.
- Checking that passwords match before account creation.

User sessions are managed via Flask-Login, which keeps users logged in across requests. Cookies are secured with the `SESSION_COOKIE_SECURE` and `SESSION_COOKIE_SAMESITE` flags to enhance security, particularly on platforms such as Hugging Face Spaces.

All authentication-related logic is handled asynchronously via JavaScript and a unified Flask API route (`/auth_action`) to improve UX while maintaining full backend validation.

Deployment URL : `https://huggingface.co/spaces/YHany/bravoai`