**Face Recognition System Documentation**

**Overview**

The Face Recognition System is a model designed to verify and identify individuals using face images and videos. It is particularly useful in a registration process where students upload their ID cards and record a video. The system performs the following key tasks:

* **Face Verification:** Compares the face from the ID card with the face from the video to determine if they belong to the same person.
* **Gender Detection:** Identifies the gender of the individual from the ID card image.

**Machine Learning Models**

1. **MTCNN (Multi-task Cascaded Convolutional Networks):**
   * **Developed By:** K. Zhang, Z. Zhang, Z. Li, and Y. Qiao.
   * **Description:** A deep learning model designed for face detection, including face alignment and bounding box detection. MTCNN is known for its high accuracy and robustness.
   * **Accuracy:** Approximately 95% in various benchmarks.
2. **DeepFace:**
   * **Developed By:** Facebook AI Research (FAIR).
   * **Description:** A facial recognition model providing comprehensive face analysis, including age, gender, emotion, and race. The gender detection model within DeepFace is highly accurate.
   * **Accuracy:** About 97% on various datasets.
3. **InceptionResnetV1 (FaceNet Framework used in pytorch):**
   * **Developed By:** Google.
   * **Description:** A deep learning model that integrates Inception architecture with residual connections, pre-trained on the VGGFace2 dataset for face recognition. It generates high-dimensional face embeddings used for face verification.
   * **Accuracy:** Over 99% in face verification tasks.

**Project Structure**

* **.env**: Contains environment variables for the application.
* **config.py**: Contains configuration settings.
* **function.py**: Implements core functionality for face processing and verification.
* **main.py**: Sets up the FastAPI application and includes routes.
* **requirements.txt**: Lists all dependencies required for the application.
* **routes.py**: Defines API endpoints and handles user authentication and file processing.

**Performance Metrics**

* **RAM Usage:** Approximately 855-860 MB.
* **Accuracy:** Approximately for face recognition 99% and gender detection 97%

**\***The accuracy of model may vary based on the quality of the image and lighting conditions

* **Processing Time:**
  + **First Time:** Approximately 40 to 55 seconds.
  + **Afterward:** Approximately 12 to 16 seconds

**Technical Details**

**Components**

1. **Configuration (config.py):**
   * **Purpose:** Manages application settings such as the secret key, JWT algorithm, and token expiration time.
   * **Features:** Loads environment variables using dotenv.
2. **Functions (function.py):**
   * **Preprocessing:** Converts image data to a usable format.
   * **Face Detection and Embedding Extraction:**
     + Uses facenet\_pytorch for generating face embeddings.
     + Utilizes MTCNN for face detection.
   * **Video Processing:** Extracts frames from video files and processes them to obtain face embeddings.
   * **Gender Detection:** Employs DeepFace for gender prediction from images.
   * **Verification:** Compares face embeddings using Euclidean distance.
3. **API Endpoints (routes.py):**
   * **Login (/login/):** Issues JWT tokens for user authentication.
   * **Face Verification and Gender Detection (/verify\_and\_gender/):** Handles file uploads, performs face verification and gender detection, and returns the results.
4. **Main Application (main.py):**
   * **Purpose:** Initializes the FastAPI application.
   * **Features:** Configures CORS, JWT authentication, includes routes, and provides a root endpoint to check if the API is running.
5. **Dependencies (requirements.txt):**
   * **Purpose:** Lists required packages including fastapi, uvicorn, opencv-python, numpy, Pillow, torch, facenet-pytorch, deepface, torchvision, passlib, bcrypt, python-jose, and fastapi-jwt-auth.

**Additional Features**

* **Error Handling:** Robust error handling for issues such as file format errors, face detection failures, and verification problems.
* **JWT Authentication:** Secures endpoints and ensures only authorized users can access certain functionalities.
* **Modular Code Structure:** Organized into separate files for configuration, core functions, and API routes, enhancing maintainability.