Smart selfie

Day 1: Development Environment Setup & Camera Configuration

Tasks:

- Install Python 3.8+ and create virtual environment for project
- Install core libraries: OpenCV, TensorFlow/Keras, numpy, matplotlib, PIL
- Install additional dependencies: dlib, face_recognition, scikit-learn
- Set up IDE (VS Code/PyCharm) with Python extensions and debugging tools
- Create Git repository and establish branching strategy
- Configure camera access permissions and test basic video capture
- Create simple camera test script to verify webcam functionality
- Set up project folder structure and documentation templates

Deliverables:

- Fully configured development environment with all libraries
- Working camera access with basic video capture script
- Git repository with initial project structure
- Environment setup guide and troubleshooting documentation

Day 2: Dataset Research & Download Strategy

Tasks:

- Research publicly available smile detection datasets (CelebA, FER-2013, GENKI-4K)
- Research age prediction datasets (UTKFace, IMDB-WIKI, AgeDB)
- Analyze dataset licenses, size, and download requirements
- Create dataset download scripts and automation
- Set up organized folder structure for different datasets
- Document dataset characteristics (size, format, annotations)
- Create dataset inventory with metadata (samples, classes, quality)

- Test sample data loading and basic visualization

Deliverables:

- Complete dataset inventory with analysis report
- Automated download scripts for all datasets
- Organized dataset folder structure
- Sample data visualization and statistics

Day 3: Dataset Download & Initial Analysis

Tasks:

- Execute dataset downloads using scripts from Day 2
- Verify dataset integrity and completeness
- Analyze smile detection dataset structure and annotations
- Analyze age prediction dataset structure and labels
- Create data quality assessment (corrupted files, missing annotations)
- Generate dataset statistics (age distribution, smile/no-smile ratio)
- Identify and document data quality issues
- Create initial data exploration notebook with visualizations

Deliverables:

- Downloaded and verified datasets
- Data quality assessment report
- Dataset statistics and distribution analysis
- Data exploration notebook with insights

Day 4: Face Detection & Preprocessing Pipeline

Tasks:

- Implement face detection using OpenCV Haar cascades and dlib
- Create face detection comparison (OpenCV vs dlib performance)
- Develop image preprocessing functions for face extraction

- Implement image resizing to standard dimensions (224x224, 128x128)
- Create pixel normalization functions (0-1 scaling, standardization)
- Implement color space conversion utilities (RGB, Grayscale)
- Create data cleaning pipeline to remove corrupted/invalid images
- Test preprocessing pipeline on sample images

Deliverables:

- Face detection implementation with comparison report
- Complete image preprocessing pipeline
- Data cleaning scripts with validation
- Preprocessed sample images for testing

Day 5: Data Augmentation & Enhancement

Tasks:

- Implement data augmentation techniques using TensorFlow/Keras
- Create rotation, flip, and brightness adjustment functions
- Implement zoom, shift, and shear transformations
- Develop augmentation pipeline with random parameter selection
- Test augmentation effects on sample images
- Create augmentation visualization tools
- Implement augmentation validation (ensure labels remain correct)
- Design augmentation strategy for smile vs age datasets

Deliverables:

- Complete data augmentation pipeline
- Augmentation visualization and validation tools
- Augmented sample datasets for testing
- Augmentation strategy documentation

Day 6: Dataset Splitting & Data Loader Implementation

Tasks:

- Implement stratified dataset splitting (70% train, 15% validation, 15% test)
- Ensure balanced distribution across smile/age classes in splits
- Create data loader classes for efficient batch processing
- Implement batch generation with on-the-fly augmentation
- Create data pipeline integration testing
- Validate data loader performance and memory efficiency
- Generate final dataset statistics for all splits
- Integrate all preprocessing components into unified pipeline

Deliverables:

- Train/validation/test dataset splits with balanced distributions
- Efficient data loader implementation
- Complete integrated data pipeline
- Final dataset statistics and pipeline documentation