AI Screen Recorder Pro 🎬

A professional screen recording application with modern AI-style interface, featuring comprehensive recording capabilities, advanced audio processing, and production-grade performance optimization.

✨ Features

🎬 Recording Capabilities

- **High-quality screen recording** with customizable FPS (15-60)
- **Multi-monitor support** with monitor selection
- **Region capture** for specific screen areas
- **Webcam overlay** with picture-in-picture positioning
- **Production-grade threading** for smooth 10+ hour sessions

🎵 Audio Features

- **Multi-device audio recording** (microphone + system audio)
- **Separate audio file export** (WAV format)
- **Real-time audio monitoring**
- **Advanced audio device selection**
- **System audio loopback** (Windows)

✨ Visual Effects

- **Mouse cursor highlighting** with customizable colors
- **Hardware acceleration** (GPU support)
- **Adaptive video quality** (Low/Medium/High/Ultra)
- **Real-time preview window**
- **Professional codec selection**

🔧 Advanced Features

- **Segment recording** for long sessions
- **FFmpeg integration** for post-processing
- **Performance monitoring** (CPU/Memory usage)
- **Modern tabbed interface** with AI-style colors
- **Professional configuration management**

🚀 Quick Start

Installation

1. **Install Python 3.8+**
2. **Install dependencies:**

```
pip install opencv-python mss numpy tkinter sounddevice soundfile pillow
psutil
```

3. **Install FFmpeg** (optional but recommended):
   - Download from https://ffmpeg.org/download.html
   - Add to system PATH or set path in application

Usage

1. **Launch the application:**

```
python main_modern.py
```

2. **Configure recording settings** in the modern tabbed interface
3. **Click "Start Recording"** to begin
4. **Click "Stop Recording"** to finish and save

📱 Modern Interface

Tabs Overview

- 📹 **Recording** - Basic recording settings and output configuration
- 🎵 **Audio & Effects** - Audio devices, visual effects, webcam settings
- 🔧 **Advanced** - Segment recording, preview mode, advanced options
- 💻 **System** - Performance monitoring and system information

Professional Design

- **AI-inspired color palette** with modern gradients
- **Responsive layout** with organized card-based interface
- **Real-time performance monitoring** with visual indicators
- **Professional button styling** with hover effects
- **Dark theme** optimized for extended use

⚙️ Configuration

Video Quality Settings

- **Low:** 480p, lower bitrate (for long recordings)
- **Medium:** 720p, balanced quality
- **High:** 1080p, high bitrate (recommended)
- **Ultra:** Maximum quality with hardware acceleration

Audio Configuration

- **Device Selection:** Choose microphone and system audio sources

- **Separate Export:** Save audio as separate WAV file
- **Loopback Mode:** Capture system audio (Windows)

Performance Optimization

- **Hardware Acceleration:** Utilize GPU for encoding
- **Segment Recording:** Split long recordings into manageable files
- **Adaptive Threading:** Automatic resource management

🛠 Troubleshooting

Common Issues

Audio not recording:

- Install audio dependencies: `pip install sounddevice soundfile`
- Check microphone permissions
- Select correct audio device in settings

Webcam blinking/unstable:

- Use webcam buffering feature (enabled by default)
- Reduce webcam update frequency in advanced settings
- Check camera drivers and connections

Performance issues:

- Enable hardware acceleration
- Reduce video quality for long sessions
- Use segment recording for 10+ hour sessions
- Monitor CPU/Memory usage in System tab

FFmpeg errors:

- Download FFmpeg from official website
- Add to system PATH or set custom path in application
- Test FFmpeg installation using built-in test button

System Requirements

- **OS:** Windows 10/11, macOS 10.14+, or Linux
- **RAM:** 4GB minimum, 8GB+ recommended for long sessions
- **CPU:** Multi-core processor recommended
- **GPU:** Optional but recommended for hardware acceleration
- **Storage:** SSD recommended for high-quality recordings

🎯 Professional Use

Production Features

- **Frame drop prevention** with adaptive timing
- **Buffer management** for smooth long recordings
- **Automatic error recovery** with detailed logging
- **Resource monitoring** with real-time alerts
- **Professional codec support** (H.264, MP4, AVI)

Workflow Integration

- **Batch processing** support with segment recording
- **Flexible output formats** for various platforms
- **Metadata preservation** for professional workflows
- **Quality assurance** with built-in validation

📞 Support

For issues or questions:

1. Check the troubleshooting section above
2. Verify all dependencies are installed
3. Test with default settings first
4. Monitor performance tab for resource usage

🔄 Updates

The application includes automatic dependency checking and provides helpful error messages for missing components. Keep dependencies updated for best performance.

🎬 **AI Screen Recorder Pro** - Professional screen recording with modern design and production-grade features.

Project Structure

New Modular Architecture (Recommended)

```
GUIScreenRecorder/
├── app_new.py            # New modular entry point ⭐
├── src/                  # Modular source code
│   ├── __init__.py
│   ├── core/            # Core recording functionality
│   │   ├── __init__.py
│   │   ├── config.py    # Configuration management
│   │   ├── audio.py     # Audio recording engine
│   │   └── video.py     # Video recording engine
│   ├── ui/              # User interface
│   │   ├── __init__.py
│   │   └── main_window.py  # Main application window
│   └── utils/           # Utility functions
│       ├── __init__.py
│       └── helpers.py   # Helper functions and FFmpeg tools
```

Legacy Version (Still Available)

```
├── app.py                 # Legacy monolithic version (working)
├── install_ffmpeg.py      # FFmpeg installation utility
```

Installation

Requirements

- Python 3.12 or higher
- Windows, macOS, or Linux

Core Dependencies

```
pip install opencv-python mss numpy
```

Optional Dependencies

```
# For audio recording
pip install sounddevice soundfile

# For mouse highlighting
pip install pyautogui

# FFmpeg (for audio/video merging)
# Download from https://ffmpeg.org/download.html
# Or use the included installer: python install_ffmpeg.py
```

Quick Start

Using the New Modular Version (Recommended)

```
python app_new.py
```

Using the Legacy Version

```
python app.py
```

Usage

1. **Set Output Path**: Click "Browse" to choose where to save your recording
2. **Configure FPS**: Set desired frame rate (recommended: 20-30 for most uses)
3. **Choose Capture Area**:
    - Monitor 0 captures all screens
    - Check "Capture specific region" for custom areas
4. **Enable Features** (optional):
    - Mouse highlighting with customizable appearance
    - Audio recording (microphone or system audio)
    - Webcam picture-in-picture overlay
5. **Start Recording**: Click "Start Recording"
6. **Stop Recording**: Click "Stop Recording"

Advanced Configuration

Audio Setup

- **FFmpeg Path**: Set the path to FFmpeg executable for audio merging
- **Device Selection**: Choose specific microphone or use system audio loopback
- **System Audio** (Windows only): Capture system sounds and music

Video Options

- **Monitor Selection**: Record specific monitors in multi-monitor setups
- **Region Capture**: Record only a portion of the screen
- **Preview Mode**: See what's being recorded in real-time

Performance Tuning for 10-Hour Recording

- **FPS Settings**: 15-20 FPS for extended sessions (balance quality vs. file size)
- **Region Capture**: Smaller regions reduce processing overhead
- **Memory Management**: The modular architecture provides better memory efficiency
- **Disable Overlays**: Turn off mouse/webcam overlays for maximum performance

Modular Architecture Benefits

Performance Improvements

- **Memory efficiency**: Better garbage collection with isolated modules
- **Threading optimization**: Dedicated threads for capture, processing, and encoding
- **Resource management**: Automatic cleanup prevents memory leaks during long recordings

Better Organization

- **Separation of concerns**: Each module handles specific functionality
- **Easier debugging**: Issues isolated to specific components

- **Code reusability**: Modules can be used independently

Maintainability

- **Simpler updates**: Modify features without affecting the entire codebase
- **Better testing**: Individual components can be tested in isolation
- **Cleaner documentation**: Each module is self-contained and documented

Troubleshooting

Common Issues

FFmpeg not found

- Download from https://ffmpeg.org/download.html
- Set path in UI or add to system PATH
- Use "Test" button to verify installation

Audio recording fails

- Install: `pip install sounddevice soundfile`
- Check device permissions
- Try different audio devices

Poor performance during long recordings

- Lower FPS setting (15-20 for 10+ hour sessions)
- Use region capture instead of full screen
- Disable unnecessary overlays
- Close other applications
- Use the new modular version (app_new.py) for better memory management

Video file corrupted

- Ensure sufficient disk space (estimate ~1GB per hour at 1080p/20fps)
- Check output path permissions
- Try different output location

Performance Tips for Extended Recording

Memory Management

- The modular architecture (app_new.py) provides better memory efficiency
- Use region capture for smaller memory footprint
- Monitor system resources during recording

Storage Planning

- Plan for ~1-2GB per hour depending on resolution and FPS
- Use fast storage (SSD) for best performance
- Consider disk space for temporary audio files during processing

System Resources

- Close unnecessary applications
- Use lower FPS for extended sessions (15-20 FPS)
- Monitor CPU temperature during long recordings

macOS Permissions

On first use, macOS will prompt for Screen Recording permission. Grant it under:
System Settings → Privacy & Security → Screen Recording

Version History

- **v1.0.0**: Modular architecture release with 10-hour recording optimization
- **v0.9.x**: Legacy monolithic version (still available as app.py)

Contributing

The modular structure makes contributions easier:

1. Identify the relevant module (core, ui, utils)
2. Make focused changes to specific functionality
3. Test individual components
4. Submit pull requests for specific modules