Project Report: Emotion-Triggered Interactive Study Aid Extension

Current Status (As of 19th July 2025)

I've developed a project that works as a browser extension aimed at improving study efficiency. The idea is that while studying (watching a video or reading), the webcam continuously monitors the user's facial expressions. If a negative emotion is detected (anything other than *happy* or *neutral*), the system captures the screen, extracts the topic being studied, and immediately pops up 2–3 MCQs related to the topic to reinforce learning and bring focus back.

CNN Model Development:

I trained a custom CNN using Keras with the following structure: two Conv2D + MaxPooling layers, followed by Flatten → Dense → Dropout → Dense layers. I used EarlyStopping and ModelCheckpoint to prevent overfitting.

- The model classifies 7 emotions: `angry`, `disgust`, `fear`, `happy`, `neutral`, `sad`, `surprise`.

- Final validation accuracy is ~52%. `Happy` and `Surprise` classes perform well, but there's poor recall for `Disgust`, `Fear`, and `Sad`, as seen in the classification report and confusion matrix.

Webcam Emotion Detection & Trigger:

I wrote a script to capture webcam frames every few seconds (interval is user-defined). If a negative emotion is detected, a screenshot of the current screen is taken.

OCR & MCQ Generation:

- I integrated the `nanonets/Nanonets-OCR-s` model from HuggingFace to extract keywords/topics from the screenshot.

- These extracted keywords are then passed to an MCQ-generation API. Although the logic works, I currently can’t run the APIs due to payment/authentication issues — exploring open-source or free APIs instead.

Web Interface:

- A minimal HTML page has been built that fetches questions from a `mcqs.json` file.

- The user has to answer 3 MCQs before the quiz window closes automatically. This simulates a real-time intervention to re-engage the learner.

Ongoing Work:

- Optimizing webcam and screen capture logic (right now, it's a bit heavy).

- Finalizing OCR and MCQ generation backend (either deployable locally or use alternatives).

- Preparing for final deployment: packaging everything into a Chrome extension or local desktop tool using Electron or Flask.

What's Next:

- Deployment and testing.

- Model improvement (I might use a pretrained model like MobileNet or EfficientNet).

- Possibly add voice input or speech recognition as a second layer of interaction.

This project brings together ML, OCR, real-time user engagement, and web technologies — and although a few components are still under integration, the core functionality is working as planned.