Low Fidelity Prototype Report

Grupo 1 - Mafalda Dinis e Martinho Figueiredo

Key components of the system

This report outlines a preliminary selection of the modules for the architecture of our application. It highlights key components such as the HTML5 API for sensor data acquisition, a sound classification model, as well as the first iteration of the rule-set enabling a context-aware user experience. These modules are chosen to ensure robust data collection, processing, and adaptability to user context, enhancing the application's functionality and user experience. As the project progresses, further refinements and specializations, such as transfer learning for the sound classification model, will be implemented based on future practical class assignments and results.

HTML5 for sensor data acquisition

HTML5 provides a robust framework for sensor data acquisition in context-aware multimedia applications. Its built-in support for various APIs, such as the Geolocation API, Device Orientation API, and Ambient Light API, allows seamless integration along with access to device sensors. This facilitates real-time data collection and processing, enhancing the app's ability to adapt to user context and deliver a more personalized experience. Additionally, HTML5's cross-platform compatibility ensures that the application can run smoothly on different devices and operating systems, broadening its accessibility and usability. Furthermore, HTML5 can access the microphone facilitating the use of a sound classification model, which will need to be implemented.

Sound Classification Model

After doing a preliminary analysis, we decided we will try to make use of a pre-trained model, trained on the ESC-50 dataset, in order to be able to implement the environmental sound recognition portion of the project. Depending on the progress and results, we will look more into transfer learning to specialize our model. This will be easier to specify once we are further along in the practical class assignment.

Decision-making and reaction - Context-Aware User Experience

Noisy Environment (Traffic, Music, etc.)

- Reduce video quality (resolution or bitrate)
- Increase audio volume

Quiet Environment (Silence, Nature sounds, etc.)

- Enhance video quality
- Balance audio volume

Speech Detected (Conversation, Meetings, etc.)

- Pause the stream
- Enable subtitles

In a noisy environment, reducing video quality and increasing audio volume ensures that the user can still follow the content without being distracted by background noise. This adjustment helps maintain the user's engagement and comprehension of the multimedia content. In a quiet environment, enhancing video quality and balancing audio volume provides a more immersive and enjoyable viewing experience, taking advantage of the serene surroundings to deliver high-quality visuals and clear audio. When speech is detected, pausing the stream or enabling subtitles allows the user to focus on the conversation or meeting without missing important information from the multimedia content. This context-aware adaptation enhances the overall user experience by dynamically adjusting the system's behavior to suit the user's current environment.

Adaptive Streaming System

To implement adaptive streaming, we plan to use FFmpeg to generate a master playlist with multiple quality levels. FFmpeg is a powerful multimedia framework that can handle video, audio, and other multimedia files and streams. By leveraging FFmpeg, we can create a master playlist that includes different quality renditions of the video, allowing the application to switch between them based on the user's network conditions and device capabilities.

For the client-side implementation, we will use the HLS.js library. HLS.js is a JavaScript library that enables HTTP Live Streaming (HLS) in browsers that do not natively support it. By integrating HLS.js, we can ensure smooth and adaptive streaming of multimedia content, providing a better user experience regardless of the user's network conditions.

This combination of FFmpeg and HLS.js will allow us to deliver high-quality, adaptive streaming content to users, ensuring optimal performance and user satisfaction.

Real-Time processing

We are currently unaware if the sound classification model will run in real time or not. Either way the system will still be able to analyze the audio environment and make necessary adjustments, albeit at a potentially slower rate. This means that while the adaptation to changes in the user's environment might not be instantaneous, the application will still provide the content as close to real time as possible.

User Interface

To provide a seamless user experience across multiple platforms, we will develop a basic user interface using HTML and JavaScript. This approach ensures that the application remains lightweight and accessible, leveraging the browser's capabilities to deliver a consistent and responsive UI. By utilizing standard web technologies, we can achieve broad compatibility and usability, allowing users to interact with the system effortlessly on various devices and operating systems. This is very compatible with the first step of sensor acquisition, as it allows for easy integration and real-time interaction with the collected sensor data.