**--Computer Science 3307A - Object-Oriented Design and Analysis 2021-2022 Fall Semester**

**University of Western Ontario, London ON, Canada, N6A 3K7**

**Magic Mirror**

**Group 20:**

Nathan Dinatale

Darwin Shiyi Liao

Peter Nicolaas Meijer

Nolan Morris

Yifei Zhang

**Instructor: Mike Katchabaw**

**1. Description**

The Magic Mirror is an overlay software for mirrors / screens that displays a variety of information such as news, weather, schedule, time, email inbox and more. The mirror uses a heads-up display to present the information to the user while simultaneously retaining it’s function as a mirror. All of the information is displayed on an LCD screen behind glass that functions as a mirror, or a webcam is used to capture video of the user to display on the screen giving the monitor a mirror-like functionality.

**2. Mandatory Features**

**2.1 User Interface**

The main feature of the smart mirror will be a configurable user interface that displays the data from the various data feeds / api integrations. The configurability will be centered around the features that are displayed (e.g. choose to display weather on page), and the location that the data will be displayed (e.g. where on screen).

**2.2 News API Connection**

QT has a variety of widgets and GUI options that are easily configured and made in its designer. Widgets such as a scrolling list, and text input boxes can be easily created. Combining this alongside the use of either Bing News Search API provided through Microsoft Azure or some third party Google News Search API (As Google had deprecated their own News API 10 years ago), it is possible to make a simple yet effective news feed that could show the most recent news. Additionally, depending on which API is used and if it can work well with the rest of the system, multiple features could be added to this news feed. Options such as personalized news preferences, advanced filters, and bookmarks could all be implemented into the news feed.

**2.3 Weather API Connection**

In regards to displaying the weather on our application, there are several options that offer varying degrees of implementation difficulty and complexity. Qt’s own “Weather Info” offers the easiest solution, and will be sufficient to display basic information such as current temperature, weather conditions and the forecast. Of course this will be localized to the city the device is currently located in, and this feature is offered by the Weather Info resource. The example provided on Qt’s website uses the weather data provided by the Open Weather Map API. The Open Weather Map API provides a plethora of weather related information even including statistics such as air pollution and the UV index. If we want to access more detailed information to use on our device, we can easily build upon and customize what Qt has provided us as we see fit. The display of the information as well will be made suitable for the application in order to synchronize with the other information displayed.

**2.4 Google Calendar API Implementation**

There are many REST APIs that help achieve the communication between software and Google products, the third party ones are easier to use, in terms of the network communication, ready-to-use functions as well as reading JSON files. However, some of the libraries aren’t free. As for Google’s own API, since none of their API client libraries are written in C++, the only way to access Google Calendar would be by HTTP calls, there are many library options that support network communications such as: Qt, Cpp-httplib, Poco, Curlpp, neon. Any of them would meet our needs, so Qt would be a good option because it is heavily related to our project. Overall, it is possible to control and interact with Google calendar on coding level as one would on their personal device.

**2.5 Date and Time Implementation**

C++ has built-in functions to access date and time, or even convert them to different time zones. QT also has additional GUI widgets such as an analog clock/digital clock that can be used instead of manually creating these elements.

**3. Optional/Wishlist Features**

**3.1 Google Voice Recognition Implementation**

Qt offers audio recording classes that can store audio in files that can then be processed by Google’s Speech-to-Text API. The connection with the API can be established using an HTTP request that is also supported by Qt libraries. The returned JSON file can then be read and processed by Qt. The voice commands can then be used to provide a variety of features for the magic mirror including interaction with the Calendar and Gmail applications to retrieve more information. Moreover, it can also be used to interact with Google's search engine directly from the mirror.

**3.2 Gmail API Implementation**

Gmail integration with the smart mirror would be a subset of features that include receiving emails and notifications, reading out emails by voice, and replying to emails (dependent on the voice recognition in 3.1 being implemented). Emails received to a configured inbox would appear on the mirror as a banner notification, with a sound effect played to alert about the delivery. The user can then use their voice to choose to read out the email and respond if desired. Enabling this feature will require compatibility with Google authentication to securely access the inbox and mail deliveries. There are various deprecated C++ libraries to communicate with the google API, a workaround might be required where Node.js is used or another language to access the API which can then interface with the C++ side application.

**3.3 User Accounts**

The user account feature will be similar to a netflix account, where a profile can be selected by the user; the information will then be configured to the user. Different mail inboxes can be configured, different news interests, and different UI configuration settings (e.g. location on mirror of weather, location on mirror of news feed, etc.).

**3.4 Lighting**

Possibilities involving remote controlled light options with voice or commands. Very Optional and not mandatory.

**4. Risks**

Working to interface modern REST API’s that are not actively supported in C++ poses a risk in that the feasibility of bootstrapping a solution is unknown, as is the time it will take to implement these features. Large sets of features in the application may become obsolete if we cannot structure a solution to a given API / data feed that we require. This means the timeline of our software project is hard to accurately predict, and must be revised throughout our development process to reevaluate the scope of our software.

**5. References**

Gnews: <https://gnews.io/>

Google Calendar API Overview: <https://developers.google.com/calendar/api/guides/overview>

Google Speech-to-Text API: <https://cloud.google.com/speech-to-text>

Google Speech-to-Text API with C++: <https://pkoretic.medium.com/making-your-own-assistant-with-qt-qml-and-google-speech-to-text-6ade0a01f731>

QT: <https://www.qt.io/>

Qt networking: <https://doc.qt.io/qt-5.15/topics-network-connectivity.html>

Qt weather info: <https://doc.qt.io/qt-5/qtpositioning-weatherinfo-example.html>

Microsoft Azure: <https://www.microsoft.com/en-us/bing/apis/bing-news-search-api>

OpenWeatherMap API: <https://openweathermap.org/api>