Google Calendar for MagicMirror: Requirements Specification

Team 8

September 19, 2016

Ethan Gill
University of Kentucky
ethan.gill@uky.edu

Arthur Silveira
University of Kentucky
arthur.silveira@uky.edu

Ethan Smith
University of Kentucky
ethan.smith1@uky.edu

Matthew Turner
University of Kentucky
matthew.turner813@uky.edu

Customer: Bill Danhauer

Table of Contents

Introduction

The increasing presence of computers in day to day life has resulted in some analog devices becoming nearly archaic. Calendars, for instance, have widely been replaced or supplemented by programs such as Google Calendar and iCal Calendars. These technologies allow users to easily access their schedules and share them with coworkers, friends, and family to coordinate their daily routine. With a new open-source project named MagicMirror, users can view their calendars, as well as news and weather information, right on their bedside or living room mirror.

This document details a software development project involving improvement of the foundation of MagicMirror's calendar module. Prerequisites of this process include setting up development and production environments for the project, including a build on a Raspberry Pi microcomputer. This will allow for the program to run on a dedicated device in order to mimic the traditional functionality and placement of an analog calendar. The Raspberry Pi also functions as a spatially efficient alternative to a desktop or similar device. This project's primary goal involves increasing the functionality of the MagicMirror program to more seamlessly interact with Google Calendars.

The scope of this project necessitates contributing to an open-source repository to develop new functionality for the MagicMirror. One core piece of the intended additions consists of allowing the MagicMirror to display shared Google Calendars, as it can currently only access ones owned by the current user. Additionally, the implementation of a notification system is desired to show the user their upcoming calendar events on the main dashboard of the MagicMirror. Moreover, this project aims to display Google Calendars and iCal Calendars alongside one another in one window. All project functionality will be designed as a drop-in module for the MagicMirror open-source repository, which currently exists on GitHub.

The purpose of this project is to access the information that MagicMirror typically displays, using a Raspberry Pi environment. The focus mainly centers on the calendar functionality; specifically, the majority of the project involves adding functionality to MagicMirror's interactions with Google Calendars. The scope includes both personal and shared Google Calendars. In short, the project should have the ability to notify the user about their upcoming events, and to merge Google Calendar and iCal Calendars into one display that the user can view. This project is made with the customer's wife in mind, such that she can plan her day

around her business obligations displayed using her personal MagicMirror. As a side effect of contributing to an open-source software repository, the project has the indirect audience of any current user of MagicMirror who wishes to use or modify this functionality.

Project Overview

Currently, the MagicMirror product includes a calendar module. This module is able to display multiple calendars merged together. However, it can only display calendars that can be downloaded in the iCal calendar format. Users who store their calendars in Google Calendar format must maintain ownership of their calendars and make them public; this is not always desirable. Additionally, MagicMirror must re-download the entire iCal file for each calendar every time it wants to update the displayed list of events, which by default occurs every five minutes.

The project described in this document will remove this constraint by leveraging the Google Calendar API to improve the MagicMirror system's usage of Google Calendars. It will also maintain UI/UX and feature parity with the old calendar module.

Customers, Stakeholders, and Intended Users

The specific customers for this project are Bill Danhauer and his wife. In general, all consumers of the MagicMirror product interested in the provided calendar integration features represent customers. The stakeholders for this product are the project implementers and the customer Bill Danhauer. The intended users of this system are people who use MagicMirror's calendar feature, specifically its existing Google Calendar integration.

Current System

MagicMirror is a full featured application that shows daily information in an aesthetically pleasing manner with the intention of displaying it on a mirror. A subsection of this application, the calendar integration feature, currently downloads calendars in the iCal format and displays them. This module is able to merge multiple calendars; however, in order to do so, it must have access to each iCal calendar file. Additionally it must re-download each one every time it wishes to update the list of events.

Proposed System

MagicMirror is an open-source project. Its plugins are primarily free and open-source as well. A module with the required Google Calendar integration does not currently exist within the MagicMirror project. Because such a solution does not currently exist, it can't be bought and has to be developed.

The primary feature of the proposed system is the ability to leverage the current calendar module and Google Calendar API to provide better features for users' Google Calendars. At present, Google Calendars are required to be owned and public for them to be displayed on the MagicMirror through the use of iCal files. The new product will make use of APIs specific to Google Calendar to allow users to avoid this restriction if they specify they are using a Google Calendar. It will also integrate seamlessly with the old iCal calendar module so that no existing functionality is lost.

Constraints

Precautions must be taken to ensure user's personal information is secure when given to the proposed system. The system should maintain a consistent UI and interface with the previous product. It should also maintain feature parity with the old module; no functionality should be lost, only added. The product must maintain compatibility with all versions of software MagicMirror depends on.

Development and Target Environments

As mentioned in prior sections of this document, the existing MagicMirror project is freely available online at a Git repository. The project is largely written in JavaScript; node.js is used for dependency management. Due to these factors, MagicMirror is largely portable and can be modified and improved using a wide range of systems, editing software, and environments.

Development Environments

Several different operating systems will be utilized for this project's development environments. The majority of work will be completed on Macintosh computers running macOS Sierra. However, a Bash environment using a Debian Linux distribution will also be configured; additionally, some work will be completed using a Bash environment on ChromeOS.

All development environments will have 'nvm' and 'node' installed. Several different text editors will likely be used for development; they include, but are not limited to Atom, Sublime Text 2, Visual Studio Code, Xcode, and Vim. Because the project is contained in a Git repository hosted on GitHub, 'git' will also be installed on all development machines. Additional programs may be used to provide a GUI for 'git', including GitHub Desktop and Tower for Git.

Target Environments

The target environments for MagicMirror are defined by the original project creators. The most commonly used environment consists of a Raspberry Pi 1, 2, or 3 running Raspbian (a

variant of Debian Linux) with 'node' and 'electron' installed. However, it is important to note that certain features of MagicMirror do not function correctly on the Raspberry Pi 1, including display output using Electron. The MagicMirror project unofficially supports any operating system with a Bash environment with 'node' present. Examples of compatible systems include macOS, Linux, Unix, and Microsoft Windows 10 with the Linux Subsystem installed.

The project will be tested for target compatibility against a Raspberry Pi 3 running Raspbian. When verifying user interface elements of the project, a screen or window with a 1080p resolution will be used.

System Model

MagicMirror works on an open source, modular plugin system, and it utilizes Electron as an application wrapper. Because of the nature of this model, MagicMirror is able to take advantage of a growing list of installable modules contributed by the open-source community. Each module adds functionality to MagicMirror, like displaying emails, daily weather or news to the user. However, one downside of such a model is that there is room for redundancy; that is, the MagicMirror project has the potential to become plagued by repetitive modules which bisect features too much, leading to repetition of the implementation. In a worst case scenario, loss of certain functionalities across multiple modules can occur. With all this in mind, the work for Google Calendar improvement will be completed in a separate, new module; nonetheless, it will be integrated back into the currently existing calendar module upon completion. Ultimately, this will result in an improved module which will add functionality without any loss of existing features or redundant implementations.

Components

Currently, MagicMirror is split into several processes which rely on the node.js and Electron frameworks. One element, the core of the project, is written in JavaScript. It contains a module loader which possesses the ability to instantiate various modules specified by the user in a configuration file. Among these modules is the calendar module which will be branched, improved and ultimately merged back into the original module. On the other hand, the UI process, which is implemented with Electron, dictates how the modules are displayed to the user.

Two diagrams of the processes which make up MagicMirror are displayed in Appendix A and B. Appendix A illustrates the System Model for the initial, minimum configuration of the

project, while Appendix B illustrates the System Model with both calendar modules combined into one.

User Interaction

Before describing the user experience when using this program, it is important to first identify the actors at play when running it. First and foremost, the user can be defined as the person or persons utilizing MagicMirror and this project's module to view their calendar information. The second actor is the MagicMirror core module itself. Since the developers of the project outlined in this document are not responsible for the development of the base MagicMirror program, it should be considered an outside actor. The core MagicMirror program is responsible for displaying the dashboard information provided by this project's program as well as other information from other active MagicMirror modules.

The improved MagicMirror calendar module (with native Google Calendar support) is the last actor at play. The module is responsible for taking calendar queries from the user and providing that information to the MagicMirror to be displayed. In order to get a satisfactory understanding of the user experience when using this program with MagicMirror, each use case described below should be considered to highlight some facet of the intended functionality.

Use Case 1: Viewing an Owned Google Calendar with Persistence

The user boots up their Raspberry Pi, and opens the MagicMirror software. The MagicMirror core program then prompts the user for their login information for their Google account in order to access Google Calendars. The MagicMirror then passes that information off to the developed calendar module, which proceeds to download that account's Google Calendar events for all calendars specified in the configuration file. The Google Calendar information is then converted into the MagicMirror native format, and passed to the MagicMirror. The MagicMirror program displays this information to the user via the Raspberry Pi. The user then turns off their monitor, but leaves the Raspberry Pi on with MagicMirror running. A week later, the user turns on their monitor to see the MagicMirror still displaying the calendar information via the Raspberry Pi, without needing to enter their information once again. Any and all chances made to the calendar's events over the course of the week are displayed on the calendar without needing to download the calendar again.

A flow chart for Use Case 1 is available in Appendix C.

Use Case 2: Two Users View their iCal Calendars as well as their Google Calendars

Two users of the developed program wish to synchronize their events from multiple calendars, some (but not all) of which are Google Calendars. The two users boot up their Raspberry Pi and open the MagicMirror base program. The MagicMirror program then prompts each user for their respective login information before passing that to the developed program. The developed program then downloads all Google Calendar and iCal calendar events from the associated accounts. The developed program translates all calendar data into one data structure before passing the data to the MagicMirror base program. Next, the MagicMirror displays any configured calendars to both users. As the two users add events on any of their calendars, the MagicMirror UI updates for each user in real time.

A flow chart for Use Case 2 is available in Appendix D.

Functional Requirements

The functionality of this program mainly consists of Google Calendar interaction. Thus, the majority of the requirements for this program focus on interacting with Google Calendar's API and handling authentication for the user. At each turn, there exists checking and handling of errors that might crop up when dealing with a network connection. Handling user authentication and gracefully failing when presented with possible configuration errors are also necessities. Finally, it is expected that the program has the ability to display calendar information to the UI.

- System loads calendar module configuration object if present in MagicMirror configuration file
    - System exits without error if calendar module configuration is not present (shows message on MagicMirror UI similar to existing "no calendars")
- System determines calendar type is iCal calendar if url field is present in calendar configuration
- System determines calendar type is Google Calendar if OAuth field is present in calendar configuration
- For each Google Calendar configuration object:
    - System loads client secret from configuration if client secret is valid secret with Google Calendar API
    - System displays error if unable to access client secret from configuration object
    - System generates OAuth token from client secret if valid client secret

    - System displays UI error if unable to generate OAuth token from configured client secret
    - System downloads configured Google Calendar calendars
    - System displays error if unable to download configured Google Calendar
    - System displays error if invalid data or error is returned from downloading Google Calendar data
    - System formats Google Calendar data into MagicMirror calendar data
    - System merges all valid formatted MagicMirror calendar objects into one object
    - System displays merged calendar object to configuration specified block in UI
- For each iCal configuration object:
    - System downloads iCal file from configured URL
    - If an error is encountered, system displays error to UI
    - System parses and formats file data into MagicMirror calendar format
- System merges all configured calendars (both Google and iCal) into one object for display
- System displays merged calendar object on UI
- System re-downloads information from Google Calendar based on a user specified time interval
    - System displays error to UI if any errors occur while downloading
    - System displays new calendar information alongside unchanged information in UI

Nonfunctional Requirements

- System displays calendars on MagicMirror interface in a style synonymous with the original calendar module
- System must be compatible with existing users' configuration files
    - System accepts JSON configuration dictionaries with the same format as original calendar module
        - iCal calendar configuration must not change from current version; this allows for forward compatibility with no changes necessary
        - Google Calendar configurations need to have a flag to let our new module know they should be handled differently; they do not need to be backwards compatible
- Authentication tokens should preserve security of user login information
    - Username and password must not be input in plain text format in any configuration file
- Response/download time for a Google Calendar should be less than or equal to response time for an iCal calendar (on average)

- All code written should match the existing MagicMirror project in terms of portability
  - All node.js versions supported by MagicMirror currently should be supported by new code
    - This requirement can be ignored if a version of node.js has been listed as insecure or support for it has been dropped by node.js maintainers (this supersedes MagicMirror support)

Feasibility

This project is feasible within the scope of a semester. The Google Calendar API is well defined and well tested; it can provide the data that this application's functionality requires. The MagicMirror calendar module's configuration standard is also well defined and is currently used in production environments. The created application must bridge these two tools and help them to integrate efficiently with the knowledge that desired calendar is a Google Calendar. This bridge will not be trivial, as the two domains know nothing about each other, and it will be necessary to convert from one to the other while handling any errors that arise. The required functionality, detailed above, comprises approximately one and a half months of development time. This is a reasonable amount of work to be completed in a semester.

Minimum Requirements

The bare-bones version of this project should include the following functionality: The project interfaces with Google Calendar API, uses 2-factor authentication with OAuth to authorize users, is able to access shared and/or private user calendars, and refreshes the calendar's data at a user configurable interval. Interfacing with Google Calendars API should be trivial. Google provides extensive documentation and sample code to accomplish this task. Once authenticated, it is possible to access the required data with a simple network request.

OAuth authentication will likely prove more complicated, but should still be achievable. It requires that the user has set up an application through Google's API and provided the new calendar module its key so that the user's data can be accessed. This will be non trivial, but within the scope of the project and its timeline. If these goals are accomplished within an optimal time period, the ability to access shared or private calendars that the user has access to should already exist in the created codebase. Refreshing calendar data involves making another network request to the Google Calendar API and displaying the data it returns. This should be a trivial addition once code exists to initially download Google Calendars.

Enhancements

The enhanced version will build upon the bare-bones version by providing the following functionality: The project will combine Google Calendar and iCal calendars into one UI module and reauthorize the user so they don't have to sign in more than once. Merging the new Google Calendar data with the old iCal file data is a small problem. It will require formatting the Google Calendar to match the expected format the UI code would like it in. Re-authenticating the user's access to Google Calendar is nontrivial; it will require generating another OAuth token from their configuration and updating the system to use that OAuth token for requests instead of the expired one.

However, there may be unforeseen limitations in the Google Calendar API that prevent generating OAuth tokens in such a way, or that enforce limitations on how many times the program can use a client's application key before they must provide a new one. For these reasons, this approach may not be practical, and it may be necessary to instead engineer another workaround to OAuth expiry.

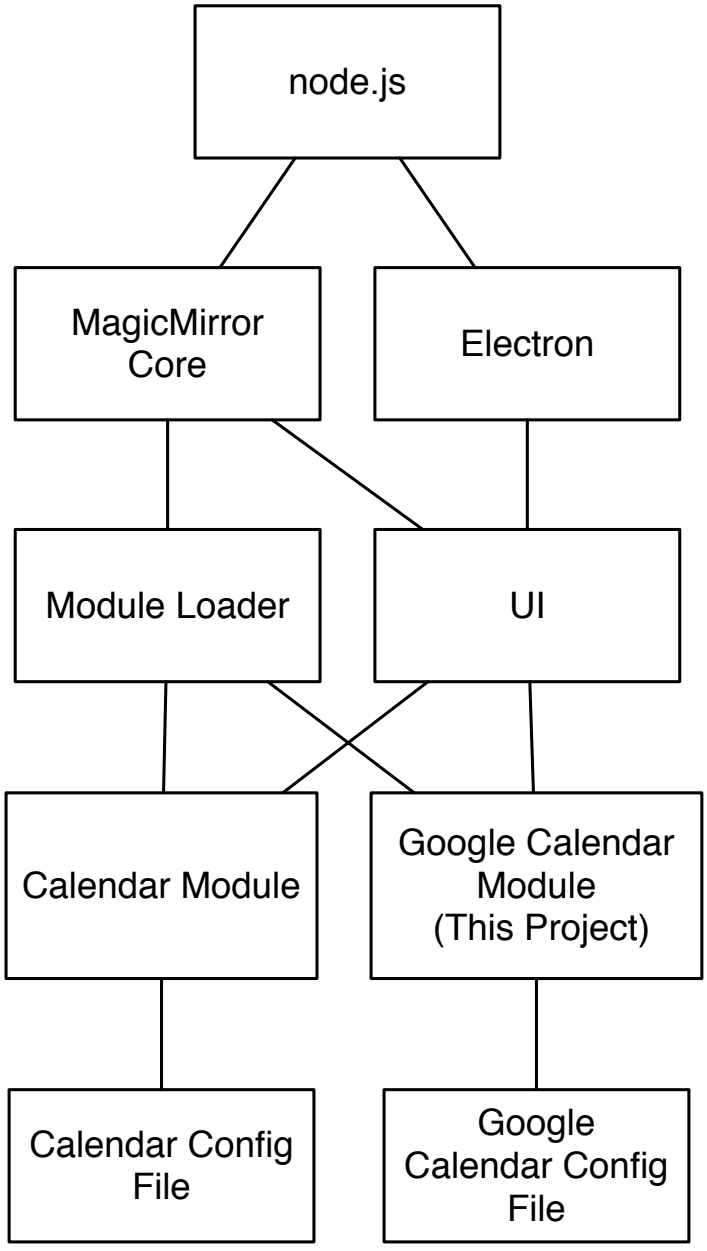
Conclusion

As of September 2016, the MagicMirror project website has almost 1,000 registered forum members. Over 2,000 people have starred the project source code on GitHub, and nearly 800 forks of the repository are publicly viewable. The potential user base for the improvements to this project detailed in this document is decently large. At this project's conclusion, the completed work will be listed as a pull request on the public MagicMirror repository to allow the original project maintainers to merge it into the greater project for all users. An open dialogue will be preserved to allow for further requirement completion if necessary to merge the additions.

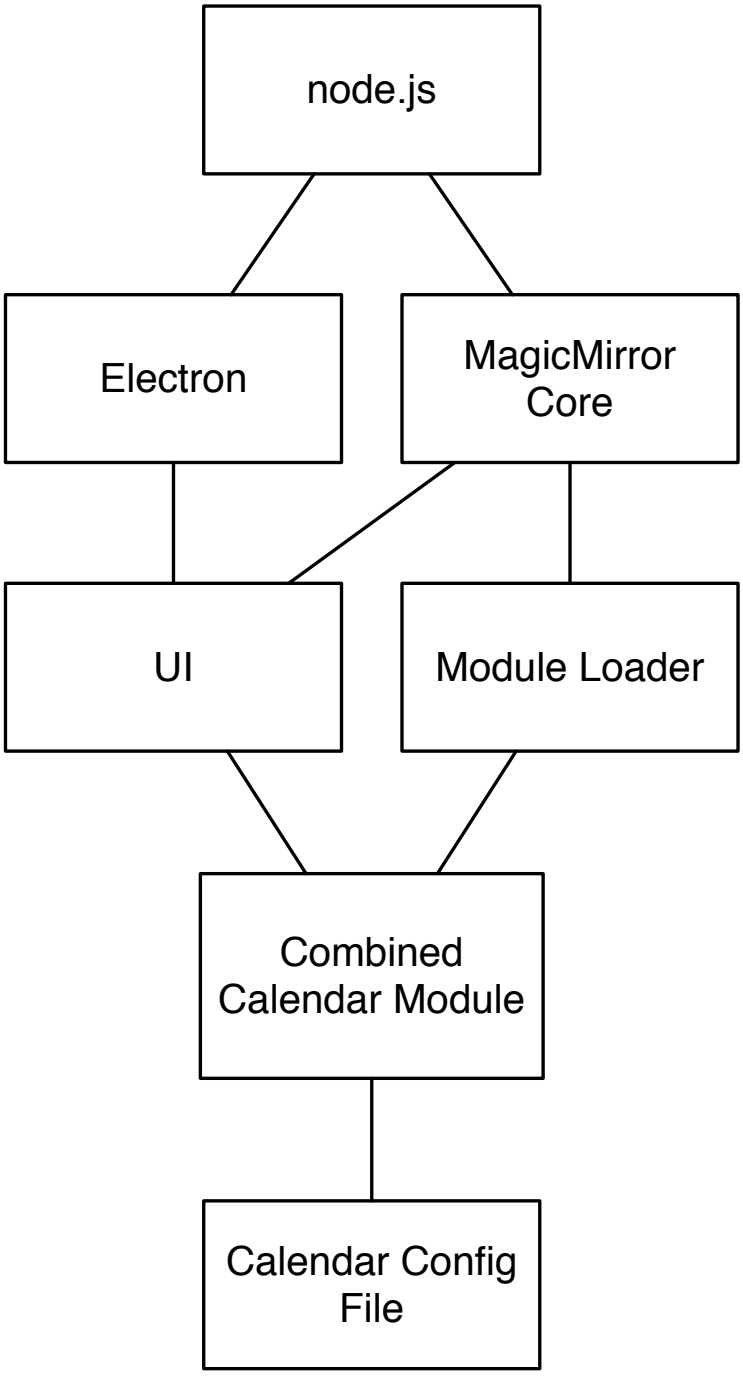
Native Google Calendar support will improve the user experience of MagicMirror as a whole, and will allow many people to use their calendars with the project who were unable to before. The benefits of this integration are immediately useful, and they have the potential to remain a part of the greater MagicMirror project for the foreseeable future.

Appendices

Appendix A: System Model for Minimum Requirements



Appendix B: System Model for Enhanced Requirements



Appendix C: Use Case 1 Flow Chart

Appendix D: Use Case 2 Flow Chart