Google Calendar for MagicMirror

Coding Assignment

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

Implementation

Source Code

The source code for the greater MagicMirror project is available on GitHub. To complete this Google Calendar project, a fork of the MagicMirror repository was created. All code written for this project can be found in this fork on GitHub, located at the link below:

https://github.com/thunderseethe/MagicMirror

User's Manual

This user's manual describes the functionality of the Google Calendar Module for MagicMirror. While doing so, it describes steps required to set up the greater MagicMirror project. However, this user's manual is not a replacement for the MagicMirror instructions, which can be found at:

https://github.com/MichMich/MagicMirror#manual-installation

Installation

Download Dependencies

This project is designed to run on Node.js and act as an extension of the MagicMirror program. Thus, the Google Calendar Module requires certain dependencies, including those of the greater MagicMirror project. Thus, the first step in installing our project is downloading and installing Node.js. Instructions to do so can be found at:

https://nodejs.org/en/download/package-manager/

Following the installation of Node.js, there are other programs and libraries which need to be downloaded, the most obvious of which is the MagicMirror base program. This can be downloaded from the MagicMirror GitHub page. If it was downloaded to the desktop, it should then be unzipped. However, users may also use the terminal 'git' command to clone the repository; if the repository is downloaded in this manner, no unzipping need be done.

Here is a link to the MagicMiror project GitHub page:

https://github.com/MichMich/MagicMirror

Ideally, this MagicMirror clone/download would contain our project. However, if the MagicMirror developers do not accept our pull request, the GoogleCalendar project fork will have to be downloaded instead. Much like MagicMirror, this can be found at this project's

GitHub page and can be cloned/downloaded from there. This following link should be used in place of the above link if this project's changes have not been merged with the master MagicMirror branch.

Here is a link to this project's GitHub page:

https://github.com/thunderseethe/MagicMirror

NPM Installations

The NPM (node package manager) is what will take these downloaded libraries and move them into the Node.js environment. In either the terminal (Mac and Linux) or the Node.js command prompt (Windows), the user should first change directories to the root folder of the project folder they downloaded. After doing so, the user should execute the following command:

"npm install"

This will create the node_modules directory and create the "magicmirror" sub-directory within "node_modules". Sub-dependencies required for Google Calendar integration will also be installed. If any error messages appear when running this command, the user should verify that they are running the command in the root directory of the downloaded project, and that they have 'npm' installed on their device.

Configuration

Before running the MagicMirror Program, the configuration file must be edited. By default, the MagicMirror code is installed with a sample configuration file "config.js.sample". If the name of this file is changed to "config.js", MagicMirror will run without any problems. However, this default file does not contain a reference to the google_calendar module. As such, this needs to be added into the file. Our project comes with a file "calendar add-in.txt" which contains the necessary information to load a Google Calendar within MagicMirror. This file is located in the google_calendar directory. For each calendar the user wishes to view via MagicMirror, they simply need paste a copy of the contents of "calendar add-in.txt" into "config.js". The user must also supply to each copy a unique "name" field and the "google_id". The "google_id" field should be filled with the email address associated with the Google Account that has access to the desired calendar.

Execution Instructions

Once the dependencies have been installed and the OAuth Token has been generated, the process for running our project is rather straightforward. Simply open up either the terminal

(Mac and Linux) or the Node.js command prompt (Windows) and change directory to the root magicmirror directory. By default this command will be:

“cd ~/magicmirror“

Once there, the user must simply run the command "npm start" and the MagicMirror application will launch with the google_calendar project as a dependency, and will begin to interact with the specified Google Calendars.

Token Generation

When the user attempts to execute the MagicMirror and google_calendar programs for the first time (with the google_calendar module included in ‘config.js’), the user will be prompted via the terminal to follow a link. Upon following this link, the user will be prompted to log into their desired Google account. The user will then be given an OAuth token which will serve as authentication in the future, eliminating the need for the user to enter their credentials in the future. The user must then feed that token (a string of characters) back to the command line so that our module can store this for the user. This process must be completed only once per user account which is specified in ‘config.js’.

Conclusion

To reiterate, much confusion can be cleared up by consulting the MagicMirror's documentation. While this User Manual is essential to understanding and using our module, it is no substitute for the documentation put out by the MagicMirror development crew, as much of our own documentation is a summation of our experience that was guided by their documentation.

Testing

Software testing plays a vital role in the developmental cycle of a project. Two main types of testing will be used to validate successful implementation of this MagicMirror project’s specifications. First, End-to-End Testing will be performed from the role of an end user to verify that the system behaves correctly from start to finish for a standard use case. Additionally, functional testing requirements outlined in this section will be used to confirm that specific functions of the created Calendar module work correctly.

Unfortunately, the current MagicMirror project does not implement any automated testing whatsoever. Because of the inherent restrictions associated with contributing to an open-source

repository, implementing automated testing for the Calendar module will prove a trying task. It will be possible to create a test data class for the user interface portion of the project, and testing the Google Calendar API code automatically may also be achievable. However, no suitable automated test options exist for the MagicMirror core project. All code modified within MagicMirror was tested manually using the functional testing requirements listed below.

Engineering Requirement Verification

Table 1 includes all project engineering requirements specified in this project's Requirements Specification.

Table 1: Engineering Requirements

Engineering Requirement	Verification	Status
System loads calendar module configuration object if present in MagicMirror configuration file	A properly configured instance of MagicMirror is started with the Google Calendar module; "no calendars" error message is not displayed	Complete
System exits without error if calendar module configuration is not present (shows message on MagicMirror UI similar to existing "no calendars")	A properly configured instance of MagicMirror is started with Google Calendar Module; "no calendars" error message is displayed	Complete
System determines calendar type is iCal calendar if url field is present in calendar configuration	A valid calendar module configuration is loaded; if the loaded object has an "url" field the system processes it as an iCal file calendar	Complete
System determines calendar type is Google Calendar if OAuth Token field is present in calendar configuration	A valid calendar module configuration is loaded; if the loaded object has an "token" field the system processes it as a Google Calendar	Complete
For each Google Calendar configuration object:		
System loads client secret from configuration if client secret is valid secret with Google Calendar API	A valid Google Calendar configuration object is present; verify that system proceeds with client secret present in this configuration	Incomplete/ Changes Made

Engineering Requirement	Verification	Status
System displays error if unable to access client secret from configuration object	An invalid Google Calendar configuration object is present; verify that error is displayed onscreen	Incomplete/ Changes Made
System generates OAuth token from client secret if valid client secret	A valid client secret is extracted from Google Calendar configuration; verify that system generates an OAuth token given a valid client secret	Complete
System displays UI error if unable to generate OAuth token from configured client secret	An invalid client secret is extracted from Google Calendar configuration; verify that system displays/logs an error signifying it is unable to proceed with configured client secret	Complete
System downloads configured Google Calendar calendars	A valid OAuth token has been generated and configured calendar exists and is valid; verify system has stored the data downloaded from configured Google Calendar	Complete
System displays error if unable to download configured Google Calendar	A valid OAuth token has been generated and network error occurs; verify system displays an “unable to reach calendar” error message to user	Complete
System displays error if invalid data or error is returned from downloading Google Calendar data	A valid OAuth token has been generated and configured calendar is invalid; verify system displays an “invalid calendar” error message to user	Complete
System formats Google Calendar data into MagicMirror calendar data	A valid Google Calendar calendar has been downloaded, verify system has valid MagicMirror data generated from Google Calendar data	Complete
System merges all valid formatted MagicMirror calendar objects into one object		Complete

Engineering Requirement	Verification	Status
System displays merged calendar object to configuration specified block in UI	Valid merged calendar data has been created; verify system displays merged calendar data to the user in configured UI slot	Complete
For each iCal configuration object:		
System downloads iCal file from configured URL	A valid iCal file URL is provided; verify system has stored data of iCal file downloaded from configured URL	Complete
If a network error is encountered, system displays error to UI	Verify system displays “unable to reach calendar” error to users	Complete
If an invalid iCal file exists at the specified URL, system displays error to UI	Verify system displays “invalid calendar” error message to the user	Complete
System parses and formats file data into MagicMirror calendar format	A valid iCal file is downloaded; verify system has MagicMirror data generated from iCal file	Complete
System merges all configured calendars (both Google and iCal) into one object for display	Multiple valid calendars are configured for module and successfully loaded; verify system creates singular calendar object containing all data from calendar instances	Complete
System displays merged calendar object on UI	Valid merged calendar data has been created; verify system displays merged calendar data to the user in configured UI slot	Complete
System re-downloads information from Google Calendar based on a user specified time interval	A valid Google Calendar has been downloaded and specified time interval has passed; verify system makes new request to Google Calendar API and feeds data into system	Complete
System displays error to UI if any errors occur while downloading	A valid Google Calendar has been downloaded, specified time interval has passed, and network error occurred; verify system displays “unable to update calendar” error message to the user	Complete

Engineering Requirement	Verification	Status
System displays new calendar information alongside unchanged information in UI	A valid Google Calendar has been re-downloaded; verify system displays new data in the configured slot on UI	Complete
System displays calendars on MagicMirror interface in a style synonymous with the original calendar module	A valid calendar configuration is loaded into module; verify system UI is not noticeably changed from original module UI	Complete
System must be compatible with existing users' configuration files	Using an originally valid user configuration, verify system handles configuration the same way the original module does	Complete
System accepts JSON configuration dictionaries with the same format as original calendar module	A valid calendar module configuration with original calendar configuration present; verify system designates config as iCal	Complete
Google Calendar configurations need to have a flag to let the new module know they should be handled differently; they do not need to be backwards compatible	N/A	Complete
Authentication tokens should preserve security of user login information	Using a valid configuration with Google Calendar configuration present, verify configuration contains no sensitive user information	Complete
Username and password must not be input in plain text format in any configuration file	Using a valid configuration with Google Calendar configuration present, verify configuration does not require a username or password to function	Complete
Response/download time for a Google Calendar should be less than or equal to response time for an iCal calendar (on average)	Using a valid configuration containing an iCal configuration and Google Calendar configuration, run several experiments to verify average loading times are similar	Complete

Engineering Requirement	Verification	Status
All code written should match the existing MagicMirror project in terms of portability	Using a valid instance of MagicMirror with the new calendar module installed, confirm it runs on all listed supported platforms	Complete
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)	Using a valid instance of MagicMirror with the new calendar module installed, confirm it runs on all MagicMirror's supported versions of node.js	Complete

It is important to note that the use of Client Secret tokens in the completed project is different from our project's original specification. This is due to the fact that the Google Developer Console allows one Client ID/Client Secret pair to be used for a virtually unlimited number of users. Using Client ID and Secret in this way greatly simplifies the configuration process for the end user of the Google Calendar module. Thus, the decision was made to generate a global Client ID and Secret.

Functional Testing

Functional tests are closely related to a project's functional requirements, often on a one-to-one basis. Functional testing was performed after each development cycle, or sprint. **Table 2** contains a list of the functional tests for the updated Calendar module of MagicMirror. Engineering Requirements, detailed in the previous section, were not considered complete unless their applicable functional tests produce the expected outcome.

Table 2: Functional Test Cases for MagicMirror Calendar Module

Test #	Config Parameters	Test Case	Success Conditions	Failure Conditions	Test Status (Pass/Fail)
1	• Client ID • Client Secret • User Token • Redirect URL	Credential authentication for a Google Calendar using Google Calendar API	Authentication is successful.	Authentication fails, and an error is displayed in the UI.	Pass

Test #	Config Parameters	Test Case	Success Conditions	Failure Conditions	Test Status (Pass/Fail)
1b	• Client ID • Client Secret • User Token • Redirect URL	Invalid credential authentication for a Google Calendar using Google Calendar API	Authentication fails, and an error is displayed in the UI.	Authentication is successful.	Pass
2	• None	Authentication without any credentials in the configuration file	A note is displayed in the UI asking the user to provide credentials.	An empty calendar module is displayed with no informational message.	Pass
2b	• Client ID • Redirect URL	Authentication without client secret in the configuration file	A note is displayed in the UI asking the user to provide credentials.	An empty calendar module is displayed with no informational message, OR authentication is successful.	Pass
3	• Client ID • Client Secret • Redirect URL	OAuth Token generation using valid credentials	An OAuth Token is generated from provided valid configuration parameters.	No OAuth token is generated. An error is displayed in the UI.	Pass
3b	• Client ID • Client Secret • Redirect URL	OAuth Token generation using invalid credentials	No OAuth token is generated. An error is displayed in the UI.	An OAuth Token is generated from provided invalid configuration parameters.	Pass
4	• iCal URL	Downloading iCal Calendar	System behaves as it did before development began; iCal is downloaded	iCal is not downloaded OR there are substantial differences in behavior from the previous version.	Pass

Test #	Config Parameters	Test Case	Success Conditions	Failure Conditions	Test Status (Pass/Fail)
5	• Client ID • Client Secret • Redirect URL	Initial download of Google Calendar	System stores full future calendar data from Google Calendar API.	System stores only partial future data from Google Calendar OR system doesn't store any calendar data	Pass
5b	• Client ID • Client Secret • Redirect URL	Network Failure during initial download	System displays network error in UI.	An empty calendar module is displayed with no informational message.	Pass
6	• Downloaded Google Calendar Data	JSON formatting of Google Calendar	Google Calendar events are converted into a JSON array/ dictionary	Google Calendar events are not converted correctly OR only some events are converted	Pass
6b	• Downloaded iCal Calendar Data	JSON formatting of iCal Calendar	System behaves as it did before development began; iCal format is converted to JSON	iCal is not formatted OR there are substantial differences in formatting from the previous version.	Pass
7	• Converted JSON Calendar Data from iCal and/or Google Calendar	Chronological display of calendar data	Calendar data is displayed in chronological order	Calendar data is displayed out of order	Pass
8	• Converted JSON Calendar Data from iCal and/or Google Calendar	Multiple calendar displays	Multiple calendars are displayed with events in chronological order, regardless of originating calendar	Calendars are displayed sequentially instead of interweaved OR only one calendar is displayed	Pass

Test #	Config Parameters	Test Case	Success Conditions	Failure Conditions	Test Status (Pass/Fail)
9	• Client ID • Client Secret • Redirect URL • Converted JSON Calendar Data from Google Calendar	Follow-up sync of Google Calendar	Google Calendar data is updated from API without re downloading entire calendar (only updated or new events are downloaded)	Entire Google Calendar is re downloaded on change, OR changes are not synced at all	Pass
10	• Client ID • Client Secret • Redirect URL • Converted JSON Calendar Data from Google Calendar	Updated Google Calendar events displayed on MagicMirror after sync	MagicMirror Calendar Module UI updates to display new/modified/ deleted events in correct chronological location	Calendar Module does not update OR module updates but new event is in wrong chronological location	Pass

End-to-End Testing

End-to-End Testing was used to ensure correct execution of a program for a general use-case. Because the MagicMirror Calendar module has only one intended user type, for the purposes of this section, a user will be interchangeably referred to as a customer.

The end-to end scenario described below in the **User/Customer** section of this document was repeated with 4 different test calendar scenarios. The scenarios are briefly listed below:

- User 1 used a single iCal calendar
- User 2 used a single Google Calendar
- User 3 used a single iCal Calendar and a single Google Calendar
- User 4 used a single iCal Calendar and two Google Calendars

User/Customer

A MagicMirror user will begin by running MagicMirror for the first time. They will be presented with the MagicMirror user interface, with the Calendar module on-screen. Because

they have not yet configured calendars to display, an informational message will be shown in place of upcoming events.

Next, the user will create or modify the calendar module configuration file. They will create one dictionary for an iCal Calendar and one dictionary for a Google Calendar. Inside the Google Calendar dictionary, they will provide their client identifier and client secret, along with a redirect URL for calendar updates. The user will then restart MagicMirror.

MagicMirror's Calendar module will parse the configuration file correctly, and will immediately begin to download iCal Calendar files. At the same time, it will attempt to authenticate with the Google Calendar API using the information provided within the configuration file. If a network or authentication error occurs with either process, an error message will be displayed in the Calendar module UI, but the other calendars will continue to be downloaded and processed.

Once all calendar data from both iCal and Google providers has been downloaded, the module will convert both types of calendar data into a single JSON structure. Unnecessary metadata will be dropped. At the end of this process, a JSON file will exist that contains all upcoming events from every valid calendar configured by the user. If a calendar cannot be parsed correctly, it will be ignored and an error message will be shown; other calendars will continue to be processed.

Finally, MagicMirror will display all upcoming events from the converted JSON object onscreen in the Calendar section. The user can add or delete events from any connected Google Calendars or iCal public calendars using other calendar software. After MagicMirror syncs the calendars again, they will display updated information.

Technical Metrics

Estimated Story Points

- Setting up the Development Environments (1 points)
- Setting up the Raspberry Pi Environment (2 points)
- Sample Project using Google Sample Code (8 points)
- Google Calendar User Story
    - Deciding on format and Creating Specifications for Google Calendar config file (5 points)
    - OAuth with Google Calendar (10 points)
    - Downloading Google Calendars Initially (20 points)

    - Syncing Google Calendars (10 points)
- Merge iCal and Google Calendar User Story
    - Ensuring Cohabitation of iCal and Google Calendar Config Files (10 points)
    - Merging UI Modules (10 points)
    - Merging Backend Modules (20 points)
- Trying to Merge with the MagicMirror Source Code (13 points)

Actual Lines of Code

A total of 1032 lines of code were written for this project. A breakdown of line count by file is listed below. Paths are relative to the 'google_calendar' directory.

- 22 lines: ./calendar.css
- 251 lines: ./calendarfetcher.js
- 400 lines: ./google_calendar.js
- 104 lines: ./googlecalendarfetcher.js
- 119 lines: ./node_helper.js
- 20 lines: ./package.json
- 5 lines: ./quickstart/calendar-nodejs-quickstart.json
- 10 lines: ./quickstart/client_secret.json
- 97 lines: ./quickstart/quickstart.js
- 4 lines: ../../package.json

Complexity Analysis

Project tasks are displayed with a complexity rating in **Table 3** below.

Simple	Moderate	Complex
1. Setting up the Development Environments		
2. Setting up the Raspberry Pi Environment		
	3. Sample Project using Google Sample Code	
		4. OAuth with Google Calendar
5. Downloading Google Calendars Initially		
	6. Syncing Google Calendars	

Simple	Moderate	Complex
		7. Ensuring Cohabitation of iCal and Google Calendar Config Files
8. Merging UI Modules		
	9. Merging Backend Modules	
		10. Trying to Merge with the MagicMirror Source Code

Our module of the MagicMirror program does not have an inheritance tree. As a web-app, we did not employ object oriented design that would warrant such behavior necessitating an inheritance diagram. Below are the descriptions of each task in a bit more detail than the confines of the table would allow:

1. Setting up the Node.js environment on each of our machines such that MagicMirror and original javascript can be run.

2. Setting up the Node.js environment on a Raspberry Pi such that it can run MagicMirror when connected to a monitor.

3. Familiarize ourselves with the Google Calendar API in the Node.js package by doing their sample exercises.

4. Ensuring that OAuth triggers to authenticate the user or prompt them for authentication upon start-up of MagicMirror.

5. Confirm that we can download Google Calendar events without necessarily being in the MagicMirror module/running through MagicMirror's UI.

6. Sync up the Google Calendar downloads with the MagicMirror program.

7. Ensure that Google Calendar and iCal events can both be downloaded from the same configuration file.

8. Merge the Google Calendar and iCal after download such that both can be displayed at the same time.

9. Merge the two's modules into one consistent form that is intuitive and appealing to the user and the MagicMirror developers.

10. Try to merge our project into the main MagicMirror repository on GitHub.

Overall System Complexity

This system has no inheritance due the nature of Java Script and its lack of an object-oriented infrastructure. Therefore, coupling between object classes is also not applicable since the three classes being worked with work very much decoupled from each other.

Product Size

This product has 2 user stories. There are a total of 15 unit test cases, which can be found in **Table 4.** There are 3 classes with around 34 methods.

The current product size can further be estimated by using Product Effort, described in the next section, or by number of Story Points, as described in the Estimated Story Points Section above.

Product Effort

The product effort in terms of hours for this project has been broken down in **Table 4.** The “Bug Fixes, Client Changes” row has been estimated to include work that will be done later this semester in the Project Maintenance phase.

Table 4: Product Effort

Activity	Number of Hours
Environment Setup	20
Coding	50
Product Testing	20
Nonfunctional Requirement Implementation	20
Bug Fixes, Client Changes (Estimated)	50

Defects

The team members were unable to find any software defects associated with this program. However, the Requirements Specification lists MagicMirror repository merging as a stretch goal. Unfortunately, the team has not heard back from the MagicMirror maintainers with a definitive answer by the time this document was published.

Developer Notes and Website

As of November 14th, the goals described in our project's Requirements Specification have been completed. Our fork of the MagicMirror project is available at the link in the **Source Code** section of this document. Our Developer notes are available at the following link:

https://thunderseethe.github.io/CS499/

The provided URL links to our team's blog homepage. From this page, it is possible to access all deliverables for this project by clicking on their respective links. These links include, but are not limited to:

- Requirements Specification
- Architecture Document
- Architecture Keynote Presentation
- Coding Assignment Writeup (this document)
- Coding Assignment Presentation
- Link to Code (GitHub MagicMirror fork)

Additionally, links at the top of this page can be used to access Developer Note blog posts from each team member. The blog posts detail design decisions made by group members, describe accomplishments for each sprint our team completed, and record group meeting times.