**LSA CLASSROOM**

**Libsys Ltd**

**Software Requirement Specification**

**Calendar Integration**

|  |  |  |
| --- | --- | --- |
| **Version** | **Developer** | **Date** |
| 1.0 | Abhiijeet Saxena | 16-05-2020 |

**Purpose**

Calendar integration aims to provide a centralized project that can be integrated with any other product in order to utilize the features of calendar provided by google, outlook, etc. We aim to utilize the api provided by these companies in order to allow user to synchronize their calendar with calendar functionality provided in our own products. E.g. LSClassroom will show user all their events in the schedule tab. Any implementation in any product will be handled at the respective project end which will minimize changes in this project. Changes will be made only if any changes are made by respective api providers.

**Overview**

Google calendar provides extensive functionality to allow users maintain their own calendar.

An event in google calendar can be defined by the following -

1. event ID is unique within a calendar
2. Start and end date time
3. Contains other data such as summary, description, location ,status, reminder, attachements etc.
4. Two types based on occurence-
   1. Single Event
   2. Recurring Event
      1. Start and end fields
      2. Recurrence field – Array consisting of one or several reccurence rule. Reccurrence rule can contain **RRULE**, **RDATE** or **EXDATE.**
      3. **RRULE-** Defines a regular rule for repeating the event. Contains various fields to calculate the dates of occurence
         1. FREQ — The frequency with which the event should be repeated (such as DAILY or WEEKLY). Required.
         2. INTERVAL — Works together with FREQ to specify how often the event should be repeated. For example, FREQ=DAILY;INTERVAL=2 means once every two days.
         3. COUNT — Number of times this event should be repeated.
         4. UNTIL — The date or date-time until which the event should be repeated (inclusive).
         5. BYDAY — Days of the week on which the event should be repeated (SU, MO, TU, etc.). Other similar components include BYMONTH, BYYEARDAY, and BYHOUR.
      4. **RDATE-** Indicates additional dates or datetimes not covered by the **RRULE**  on which event is to occure**.**
      5. **EXDATE-**Indicates dates or datetimes when the event should not happen.
5. Two types based on time
   1. Timed Event - occurs between two specific points in time. Timed events use the start.dateTime and end.dateTime fields to specify when they occur.
   2. spans an entire day or consecutive series of days. All-day events use the start.date and end.date fields to specify when they occur. Note that the timezone field has no significance for all-day events.
6. Can have multiple attendees

Limits on google calendar events

1. The maximum size of a property's key is 44 characters, and properties with longer keys will be silently dropped.
2. The maximum size of a property's value is 1024 characters, and properties with longer values will be silently truncated.
3. An event can have up to 300 shared properties totaling up to 32kB in size (keys size + value size).
4. An event can have up to 300 private properties, totaling up to 32kB in size (keys size + value size), across all "copies" of the event.

In LSAC, user will opt to connect its google calendar to our product. Once the permission has been provided, it will be stored in order to avoid multiple authorization token. All modules that are visible in schedule panel will now be synced with google calendar i.e. for each module related to schedule, an add event will be called which will sync users data to google calendar. Any change in the event will make changes to data in google calendar.

All events that are visible in schedule panel will need to be identified so that their respective event triggers can be managed.

LSAC can either call a method or a rest api to add/fetch calendar data as is considered convenient. I think an API call would be better as it would mean a single deployment for calendar project and multi access from different project requiring calendar. Any update in the calendar project would not require updates to be delivered to other products utilizing the calendar feature.

**Functionality**

Google calendar functionality -

1. Maintain user authorization token so that the user does not need to authorize the application multiple times.
2. Add, edit, delete and fetch events in the google calendar.
3. The above mentioned methods can be both available as a callable method or REST API whichever is preferred.
4. Calendar data can be stored in a database in order to avoid full sync operation every time.

LSAC functionality -

1. Store gmail id and authorization token if user wishes to connect to google calendar.
2. Permission to user to allow which events to be showed in its google calendar.
3. View, edit, delete events in google calendar.
4. Sync option which user can use anytime to sync the data between LSAC and google calendar if it feels there is synchronization delay.

**Risk**

1. Once an event is added to google calendar from our software, say LSAC, on fetching LSAC schedule, same data will be visible twice due to the fact that schedule will fetch data from google calendar as well as LSAC database.
2. This risk can be dealt with while utilizing the api/method in respective product.
3. Performance identification needs to be made between fetching calendar data from google servers in realtime vs fetching the same data from our own intermediary server.

**Scheduling Estimates**

|  |  |  |
| --- | --- | --- |
| Milestone | Description | Man hours |
| M1 | Initialize project and create authorization interface. Create database to manage users and maintain their email Ids. |  |
| M2 | Add, edit, delete and fetch google calendar events | 8 hours including testing all cases. |
| M3 | Handling synchronization of resources.  Maintaining the fetched resources in database.  Version handling via etag for concurrent writes to an event. | 4-7 hours including testing |