Design of Personalized Calendar

Requirements based on the website originally by jagrutigoswami402
www.geeksforgeeks.org/system-design-design-google-calendar/amp/

***This personalized calendar** is a calendar application where you can manage your schedule with graphical user interface. You can use it to remember personal things, like birthdays, appointments, and events. You can also set reminders to help you remember when something is coming up soon.*

Requirements of the System:

The requirements are categorized into three: managing account, managing events, and reminding about events.

1. Managing Account:

1) Create a user account:

First, you should be able to create a user account by entering your details including user ID, name, password, email, notification channel, and whatever you think is required. Note that your user schema should include a notification channel attribute, but consider that its only allowed value is a pop-up message box. You need not allow other options like email.

- ✓ Calendar: Create a registration form including input fields for user details such as user ID, name, password, and etc.
- ✓ Calendar: Get inputs and request a registration to the database server.
- ✓ Database server: Register a new user and send a success or failure response.
- ✓ Calendar: Display the success or failure message from the registration.

2) Update a user account:

You can update your account details.

- ✓ Calendar: Request the account information to the database server.
- ✓ Database server: Query the matching user details and send them back.
- ✓ Calendar: Create an account settings form filled with current user details, get updates, and send modified user details to the database server.
- ✓ Database server: Update the user account and send a success or failure response.
- ✓ Calendar: Display the success or failure message from the update.

3) Authenticate a user:

Before querying or managing events, you should log in. Note that after the successful login, a monthly calendar view should be displayed by default that shows all events that are scheduled for the current month. Also note that you have to decide the number and format of event details to show in the calendar view. For example, you can display event name and time as shown in Figure 1.

- ✓ Calendar: Create a login form with user ID and password input fields, get inputs, and send an authentication request to the database server.
- ✓ Database server: Verify the user account, sends <u>all events to be scheduled for the current month</u> on success or otherwise send a failure message.
- ✓ Calendar: For zero or more events, display them in your monthly calendar view or otherwise show the authentication failure message.

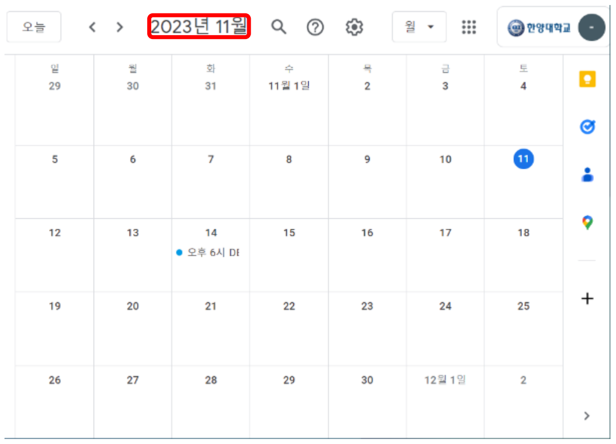


Figure 1 Example monthly calendar view

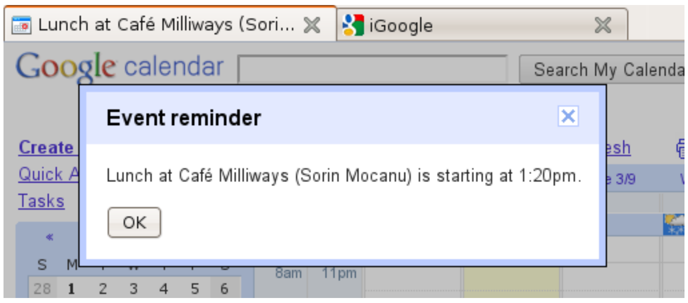
2. Managing Events:

0) About event reminders:

You can set up event reminders. Note that an individual reminder is set for each event, but <u>no default reminder setting is allowed for a user.</u>

- ✓ Each reminder should include an interval and time frame.
- ✓ The interval should specify time in minutes that is less than or equal to 60, and should be multiples of 15. That is, 0, 15, 30, 45, and 60 minutes are allowed, where <u>0 means no reminders.</u> Thus, <u>if the interval is set to 0, no time frame selection is enabled.</u>
- ✓ The time frame is a period of time within which reminders are scheduled to be sent. Similar to the reminder interval, the time frame should specify time in minutes, and should be less than or equal to 60, that is, any minutes between 15 and 60 are allowed.
- ✓ For a given event, reminders are scheduled to be sent every 'interval' minutes during the valid time frame between starting from 'time frame' minutes before the actual event time to the time just before the actual event time. Note that <u>a reminder should not be sent at the actual event time.</u>
- ✓ Each reminder will be displayed in a pop-up message box like Figure 2.
- ✓ For example, if dinner is scheduled for 6:00 PM and its associate reminder interval and time frame are set to 30 minutes and 60 minutes respectively, reminders are scheduled to be sent every 30 minutes during a time frame between 5:00 PM and 5:59 PM. Thus, reminder message boxes will be pop-up twice at 5:00 PM and 5:30 PM.

Figure 2 Example reminder message box


Event reminder

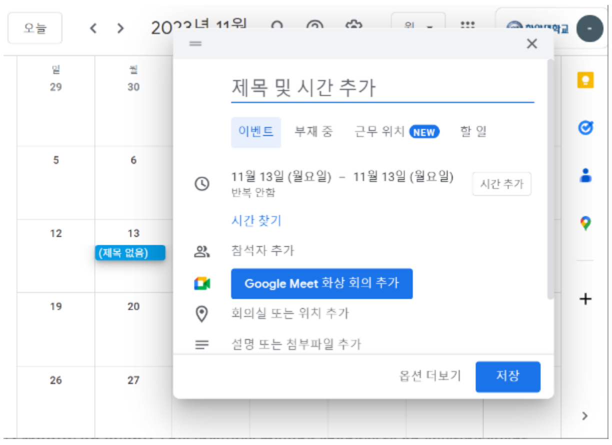
Lunch at Café Milliways (Sorin Mocanu) is starting at 1:20pm.

1) Create an event:

In a calendar, you can add an event.

- ✓ Calendar: Display an event creation form with input fields about event details.
- ✓ Calendar: Get event details and send an event creation request to the database server.
- ✓ Database server: From event details, get the target time slot, check its availability by querying events that would overlap with the requested one, and send a failure response in the case of time collision.
- ✓ Database server: If the time slot is available, insert a new event and sends a confirmation response after successful creation.
- ✓ Calendar: On success, display the event in the calendar view or otherwise show the failure message.

Figure 3 Example event creation form


11월 13일 (월요일) – 11월 13일 (월요일)
반복 안함

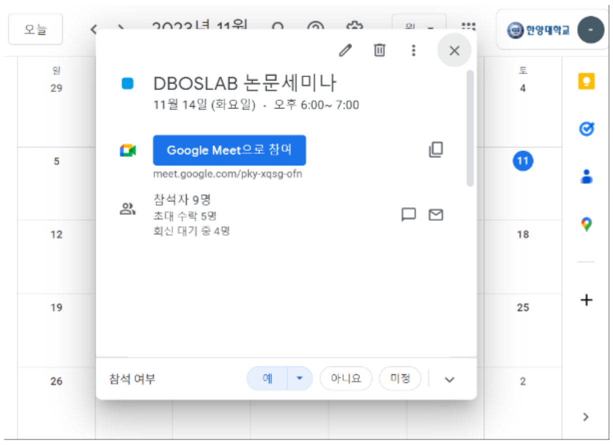
2) View an event:

This shows details about an existing event that you have created.

- ✓ Calendar: Enable a user to click on an existing event in a calendar view to query its details.
- ✓ Calendar: Send a request for querying the selected event details to the database server.

- ✓ Database server: Query the event and send its details or a failure response back.
- ✓ Calendar: Create a form that shows details of the selected event and fill it with the query results or otherwise display a failure message.

Figure 4 Example event details

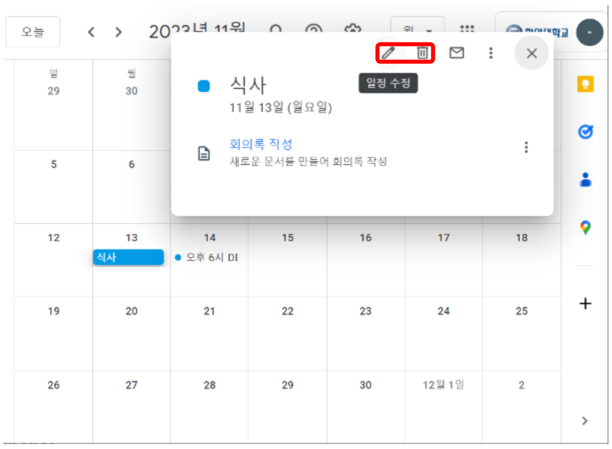

DBOSLAB 논문세미나
11월 14일 (화요일) · 오후 6:00~ 7:00
meet.google.com/pky-xqsg-ofn
참석자 9명
초대 수락 5명
회신 대기 중 4명

3) Modify an event:

This includes making changes to an existing event that you have created.

- ✓ Calendar: Enable a user to click on an existing event in a calendar view to select an edit button to open an edit form.
- ✓ Calendar: Get updates and send an event update request to the database server.
- ✓ Database server: Update the specified event and send a success or failure response.
- ✓ Calendar: Display the success or failure message and update the calendar view to reflect changes (e.g., time change) if needed.

Figure 5 Example event edit and delete buttons


식사
11월 13일 (월요일)
새로운 문서를 만들어 회의록 작성

4) Cancel an event:

This includes deleting or removing an event that you have created.

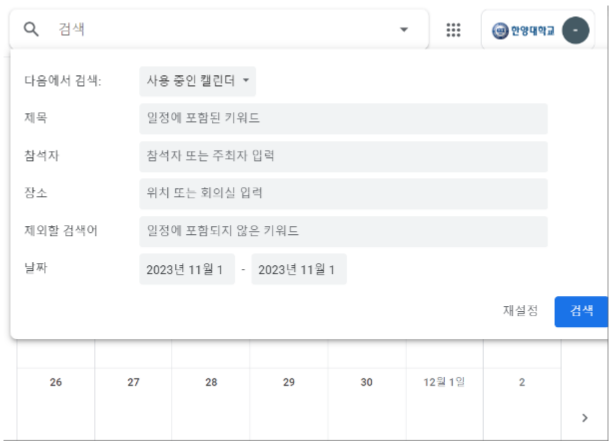
- ✓ Calendar: Enable a user to click on an existing event in a calendar view to select a delete button to cancel an event.
- ✓ Calendar: Send a delete request to the database server.
- ✓ Database server: Delete the specified event and send a success or failure response.
- ✓ Calendar: On success, update the calendar view to reflect the canceled event or otherwise show the failure message

5) Look up events in the calendar:

This includes lookup for events by search terms and viewing their list. Search terms include event details like name, location, and etc.

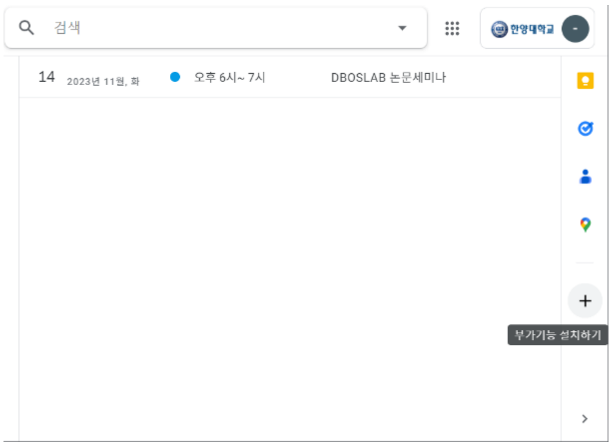
- ✓ Calendar: Create a search form with input fields for event details.
- ✓ Calendar: Get search terms and send a query request to the server.
- ✓ Database server: Query events using search terms and send the query results or a failure response back.
- ✓ Calendar: Display the search results or the failure message.

Figure 6 Example event search form



다음에서 검색:	사용 중인 캘린더
제목	일정에 포함된 키워드
참석자	참석자 또는 주최자 입력
장소	위치 또는 회의실 입력
제외할 검색어	일정에 포함되지 않은 키워드
날짜	2023년 11월 1 - 2023년 11월 1

Figure 7 List of events in the specified time



14 2023년 11월, 화 ● 오후 6시~ 7시 DBOSLAB 논문세미나

6) View calendar:

This allows you to see your scheduled events in a calendar format which can be a monthly, weekly, or daily view.

- ✓ Calendar: Create a menu with buttons for monthly, weekly, and daily views.
- ✓ Calendar: For a transition to a longer time frame, send a request of querying events in the specified time frame to the database server.
- ✓ Database server: Query events in the specified time frame and send query results or a failure response back.
- ✓ Calendar: Implement transitions between views and display query results in the changed view or display a failure message.

Figure 8 Example weekly view

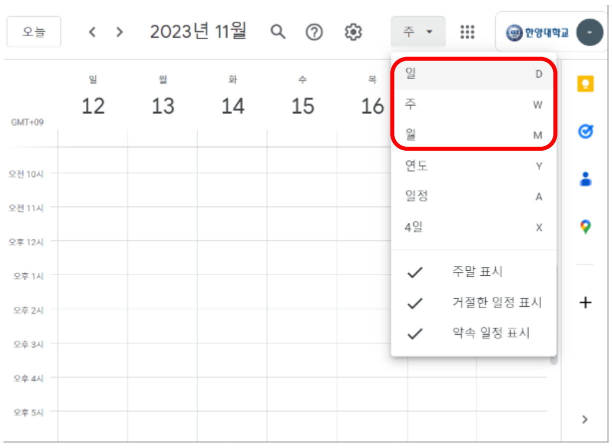
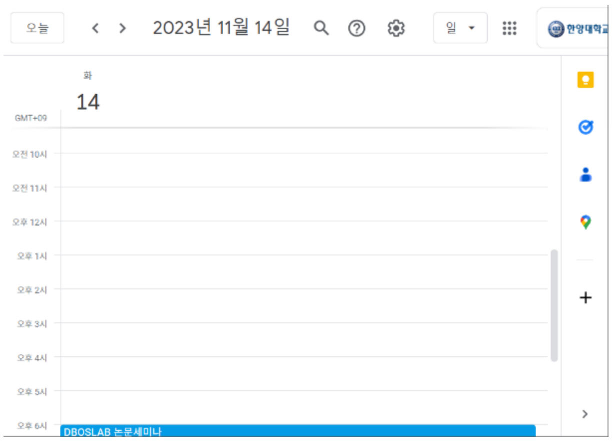


Figure 9 Example daily view



Managing Reminders:

1) Remind events:

If you set event reminders, they should be scheduled to pop-up message boxes at appropriate times specified in their associate events.

- ✓ Calendar: Create and manage a list of events whose reminders should be scheduled to pop-up message boxes, denoted as list_popup.

- ✓ Calendar: Implement a background task for querying events that will start within 1 hour but be new to the list_popup.
- ✓ Calendar: Send a request to query a list of events that will start within 1 hour.
- ✓ Database server: Query events whose reminders are set and whose start time remains less than 1 hour and then send query results or a failure response back.
- ✓ Calendar: Update the list_popup with events not included in it.

2) Display a pop-up message box:
If you set event reminders, they should be displayed in pop-up message boxes at appropriate times specified in their associate events.

- ✓ Calendar: Implement a background task for checking the list_popup and identifying events whose reminders should be pop-up now.
- ✓ For each identified event, display a pop-up message box as a reminder.

3) Delete events that have started:
If events have started, they need no more reminders and should be removed from the list_popup.

- ✓ Calendar: Implement a background task for checking the list_popup and identifying events that have started and remove them from the list.

Data Flow

Data flow for viewing the calendar:

1. The user opens the calendar application on a computer.

2. The calendar application sends a request to the database server to authenticate the user and establish a connection.

3. The database server verifies the user’s account and sends a response to the calendar application.

4. The calendar application sends a request to the database server to retrieve the user’s calendar data that is scheduled for the current month.

5. The database server retrieves the requested calendar data from the calendar database.

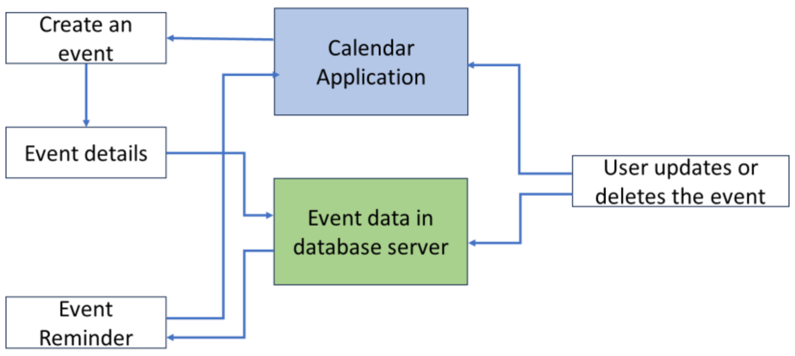
6. The calendar application displays the calendar data to the user through a monthly calendar view.

7. The user interacts with the calendar by adding, modifying, or deleting events or changing settings.

8. The database server processes the requests and updates the user’s calendar data in the calendar database.

Data flow diagram of the entire system

Figure 10 Data flow diagram of the entire system


Create an event
Calendar Application
Event details
User updates or deletes the event
Event data in database server
Event Reminder

1. The user creates an event in his or her calendar by providing event details such as the event name, date, time, description, and location.

2. The event data is stored in his or her calendar database, which is hosted on the database server.

3. Event reminders: The user can set reminders for the event based on the user's preference, for example, every 15 minutes, every 30 minutes or none. Each reminder will be displayed in a pop-up message box at a scheduled time.

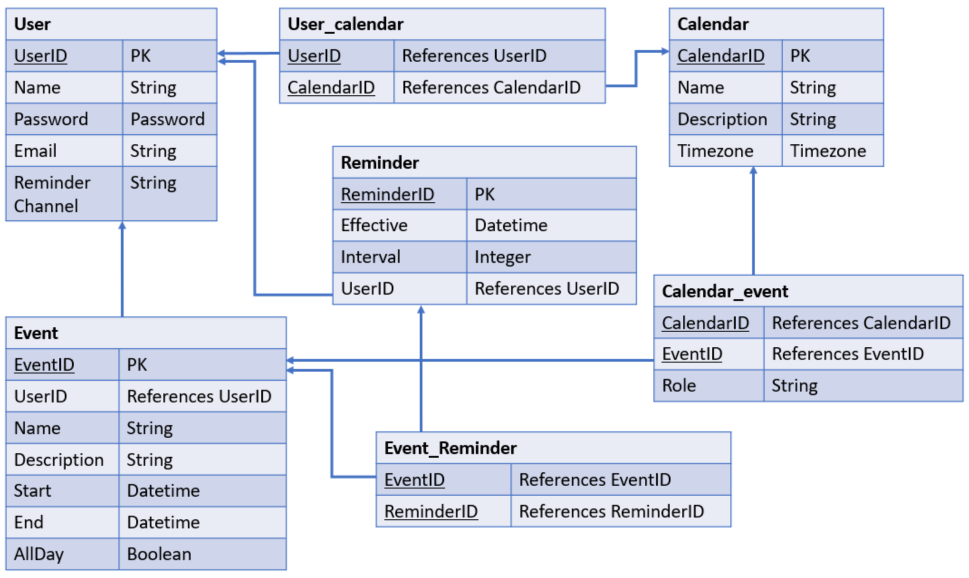
4. Event updates: The user can make changes to the event, such as changing the date or time.

5. Event deletion: The user can delete the event which will be removed from his or her calendar database.

Database schema and design

You would have the following tables to work along:

Figure 11 Database schema design



User	
UserID	PK
Name	String
Password	Password
Email	String
Reminder Channel	String

User_calendar	
UserID	References UserID
CalendarID	References CalendarID

Calendar	
CalendarID	PK
Name	String
Description	String
Timezone	Timezone

Reminder	
ReminderID	PK
Effective	Datetime
Interval	Integer
UserID	References UserID

Calendar_event	
CalendarID	References CalendarID
EventID	References EventID
Role	String

Event	
EventID	PK
UserID	References UserID
Name	String
Description	String
Start	Datetime
End	Datetime
AllDay	Boolean

Event_Reminder	
EventID	References EventID
ReminderID	References ReminderID

It is basically about understanding the requirement from a functional perspective and deciding how you are going to utilize the database and how are you going to design it.

API design of the system

Let's do a basic API design for your services:

For Scheduled Events

1. Get events for the user for the specific period.

GetEvents(userID, date, days)

2. Creating an event

CreateEvent(userID, startTime, endTime, users, subject, description)

e.g., CreateEvent(userID, startTime, endTime, ["abc", "xyz"], "Important Event", "Let's go out for dinner")

3. Updating the event

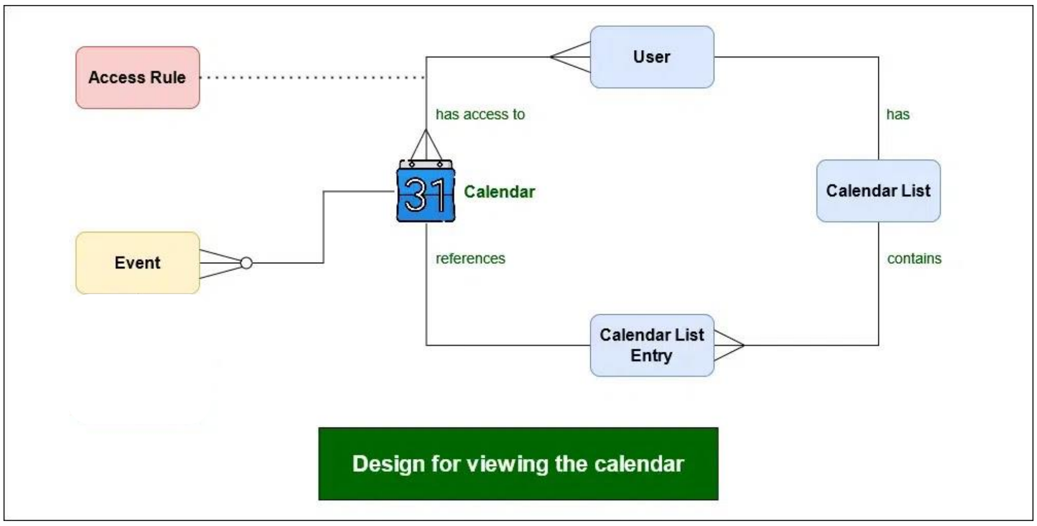
UpdateEvent(eventID, startTime, endTime, users, subject, description)

e.g., UpdateEvent(eventID, startTime, endTime, ["abc", "xyz"], "Event Cancelled", "Let's go out for dinner")

High-Level Design of the System

Design for showing the calendar and the related events to a user

Figure 12 Design for a particular user viewing the calendar and the related events to that user



<u>Conclusion</u>

In conclusion, designing a system like this personalized calendar needs careful consideration of various components and features which would be crucial to its functionality. The system must be scalable and flexible, allowing for easy integration with other applications and platforms. It should also be secure and reliable.

Overall, a well-designed system for the personalized calendar should provide users with a reliable and efficient tool for managing their daily schedules and events.