جامعة اليمامة
Al Yamamah University

AL YAMAMAH UNIVERSITY

College of Engineering and Architecture

Bachelor of Science in Software and Network Engineering

Jadwal: An Elegant, iOS-based Calendar Manager

Graduation Project

Group Project Submission

Student Names	**Student IDs**
YAZED ALKHALAF	202211123
SAIMAN TAKLAS	202021400
AFFAN MOHAMMAD	202211086
ALI BA WAZIR	202211018

Submission Date: 19 Sep 2024
Supervised By: Dr. Inaya Allah

First Semester 2024–2025

ABSTRACT

ACKNOWLEDGMENT

Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS

1 INTRODUCTION

1.1 Background of the Study

As the world is moving towards globalizing, effective time management is becoming very important. Considering how everything seems to be rushing in today's world, there is a high requirement for an effective user-friendly time management tool. Paper-based calendars have been used for addressing the complexities of managing multiple schedules across various aspects of life such as work, school, and personal commitments. However, they often fall short in providing a comprehensive solution to modern scheduling challenges.

The introduction of the digital calendar has somewhat solved this problem but still, users face a lot of issues in keeping their calendars up-to-date and synchronized. There are still few people that manually input events into their calendars. This could be really tiring, especially when dealing with multiple calendars.

Moreover, the rise of instant messaging platforms like WhatsApp has changed the way we communicate and plan events. Mostly, important dates and appointments are discussed informally leading to a disconnect between where the information is initially shared and where it needs to be recorded for effective time management.

To address these challenges, we are planning an application called *Jadwal*, which aims to revolutionize how people manage their time and schedules in the digital age.

1.2 Problem Statement

Users often face challenges in keeping their calendars up-to-date, particularly when dealing with information from various sources, including informal communication mediums like WhatsApp. The process of manually adding events to the calendar is both time-consuming and prone to errors. Additionally, managing multiple calendars—such as those for work, school, and personal life—creates further complexity and increases the risk of scheduling conflicts. The lack of seamless integration with popular communication platforms exacerbates the problem, leading to a higher likelihood of missing important events due to the scattered distribution of information across different calendars and data sources.

1.3 Objectives of the Study

The main objectives of Jadwal are:

- To develop an intelligent calendar management system that automatically extracts events from the communication channels and adds them to the user's main calendar.

- To create a user friendly interface that allows users to automatically add events to the calendar.
- To implement smart resolution system that notifies users of scheduling conflicts and provides easy options for resolution.
- To integrate all the calendars into Jadwal's single calendar view to make viewing and managing all the events easy.
- To prioritize and automatically schedule daily routines such as waking time, sleeping time and prayer time.
- To significantly reduce the time users spend on manual calendar management.

1.4 Scope of the Study

Jadwal is not just another calendar application; it's a comprehensive time management tool designed to aggregate and optimize your existing calendars and data sources. The scope of the project includes:

- Development of an iOS application as the primary platform.
- Integration with calendars using CalDAV.
- WhatsApp message parsing for event extraction (subject to technical feasibility).
- Target audience: Busy professionals, students, and anyone juggling multiple schedules.
- User testing phase to ensure ease of use and effectiveness.
  Our testing methods will include:
- Beta testing with a diverse group of users.
- Analytics to track user behavior and app performance.

1.5 Significance of the Study

Jadwal endeavours to solve problems and its significance can be summarized in the following:

1. **Time is Money**: Time is the only asset you can't get more of, it is being consumed til the last day of your life.
2. **Prayer First Calendar**: Prayer times come first, then your daily scheduled items.
3. **Streamlined Time Management**: By automatically extracting events from various communication channels, Jadwal significantly reduces the time and effort required for manual calendar management, allowing users to focus on more productive tasks.
4. **Reduced Human Error**: Automated event extraction and addition to calendars minimize the risk of missing important events or appointments due to manual input errors or forgetfulness.
5. **Integrated Communication and Scheduling**: By bridging the gap between informal communication (e.g., WhatsApp) and formal scheduling, Jadwal addresses a critical pain point in modern time management.
6. **Conflict Resolution**: The smart resolution system helps users identify and resolve scheduling conflicts efficiently, reducing stress and improving overall time management.
7. **Holistic View of Commitments**: By integrating multiple calendars into a single view, Jadwal provides users with a comprehensive overview of their commitments

across various aspects of life, facilitating better decision-making and work-life balance.

1.6 Limitations of the Study

Nothing is perfect, and our project is not a outlier. The limitations we have figured out about it are as follows:

- WhatsApp integration allows the app to read the users messages, so it would be hard to prove privacy hasn't been breached.
- WhatsApp integration might not always be there, they are a third-party.
- Learning new technologies for iOS development might require more time than anticipated.
- Accuracy of our algorithms to detect keywords indicating an event agreement has happened, especially for languages other than English.
- Time and manpower constraints may limit the number of features we can implement.
- Dependency on third-party calendar APIs and their limitations.

1.7 Organization of the Senior Project

Our project plan can be illustrated in the following gantt chart, **Figure 1.1**.

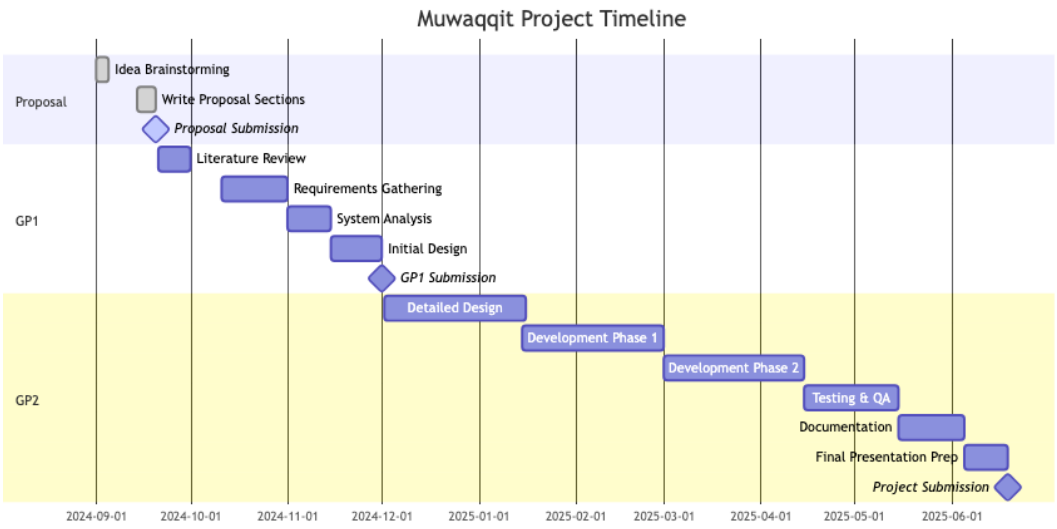

Figure 1.1: Project Gantt Chart

2 LITERATURE REVIEW

In developing Jadwal, we have drawn inspiration from and built upon existing research and products in the field of intelligent calendar management. Some key references include:

- **Clockwise (https://www.getclockwise.com/):** A smart calendar assistant that optimizes schedules and manages team coordination [Clockwise, 2024]. Clockwise's approach to intelligent time blocking and meeting optimization provides valuable insights for Jadwal's automated scheduling features.
- **Motion (https://www.usemotion.com/):** Motion's Intelligent Calendar takes your meetings, your tasks, your to-do list, your activities, and creates one perfect, optimized schedule to get it all done [Motion, 2024].
- **Reclaim AI (https://reclaim.ai/):** An intelligent time management tool that helps optimize schedules and automate tasks [Reclaim, 2024].
- **Calendi (https://calendi.ai/):** Calendi describes itself as: "Calendi is an AI calendar system. Use it for scheduling tasks, automating meetings, and witness the future of calendar." [Calendi, 2024]
- **An Exploratory Study of Calendar Use:** "Prospective remembering is the use of memory for remembering to do things in the future, as different from retrospective memory functions such as recalling past events." [Tungare et al., 2008]
- **WhatsApp Integration:** Our research indicates that direct WhatsApp integration for event extraction has not been widely implemented in existing calendar applications, making this a unique feature of Jadwal.

Feature	Jadwal	Clockwise	Motion	Reclaim AI	Calendi
Open Source	✅	❌	❌	❌	❌
WhatsApp Integration	✅	❌	❌	❌	❌
CalDAV Support	✅	✅	✅	✅	?
Conflict Resolution	✅	✅	✅	✅	?
Prioritize Prayer Times	✅	❌	❌	❌	❌
iOS Application	✅	✅	✅	✅	?

Figure 2.1: Feature Comparison Table

3 SYSTEM ANALYSIS AND DESIGN

3.1 Functional Requirements

- The user shall be able to access their account using either Google OAuth or magic link via Email. For new users, a new accout is created, and for existing users, they are given access to their account directly
- The system shall send a welcome email to new users.
- The user should be able to connect a calendar using CalDAV.
- The user should be able to connect their WhatsApp account.
- The user should be able to add events manually and set priorities optionally.
- The user should be able to view integrated calendar.
- The user should be able to configure daily routines.
- The user should be able to manage scheduling conflicts.
- The user should be able to schedule prayer times.
- The system shall send event notifications to the user.
- The system shall personalize the experience based on answers provided by the users.
- The system shall add the WhatsApp extracted events to the calendar. If a conflict occurs, the user shall get a notification to resolve the conflict with suggestions.
- The system shall synchronize calendar data across multiple devices.

3.2 Non-Functional Requirements

- **Platform Compatibility:** The app shall be compatible with iOS devices running iOS 14.0 or later.
- **Performance:** The app shall load the main calendar view within 3 seconds on 5G with speeds above 200mpbs.
- **User Experience:** The user interface shall follow iOS Human Interface Guidelines for consistency and ease of use.
- **Security:** All data transmissions between the app and servers shall be encrypted using HTTPS.
- **Localization:** The app shall support localization in Arabic and English.
- **Data Privacy:** The app shall comply with the data protection regulations and laws in Saudi Arabia.

3.3 System use-cases

Figure 3.1 shows the use case diagram for the system of Jadwal.

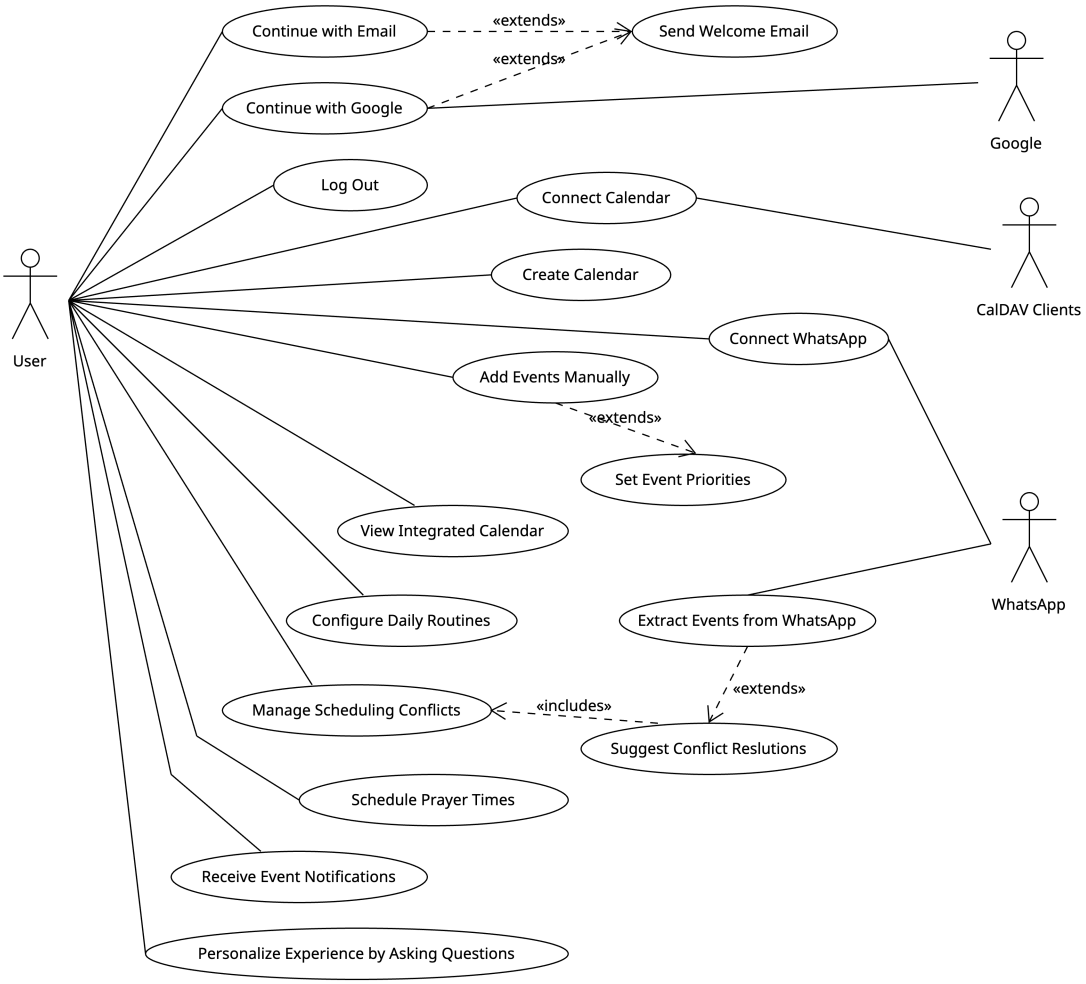

Figure 3.1: Use Case Diagram of Jadwal

Continue with Email

Basic Information

ID Number: 1 **Priority:** HIGH **Type:** Regular

Short Description

This UC allows users to login or create an account using their email.

Trigger

This UC starts when the user enters their email to the system.

Actors

Primary: User **Secondary:** None

Preconditions

User must have an email

Relationships

Extends: Send Welcome Email **Includes:** N/A **Generalization/Specialization:** N/A

Major Inputs

- **Email** (Source: User)
- **Magic Link (from email)** (Source: User)

Major Outputs

- **Magic link email** (Destination: User)
- **Confirmation messages** (Destination: User Interface)

Main Flow

1. The user enters their email.
   Information: System displays email input field.
2. System generates and sends magic link.
   Information: System sends email and displays ”Check your email” message
3. The user clicks the magic link.
   Information: The app is opened on the device of the user
4. The app sends the token to the system to log the user in
   Information: System verifies token and logs user in.

Alternate Flows

1. If the user has no account, the system creates a user.

Exceptions

- Invalid email format.
- Email not registered (if login only).
- Magic link token expired or invalid.

Conclusion

This UC ends when the user is logged in.

Post-conditions

The system generates a JWT.

Special Requirements

An email server must be present to send email magic link.

Continue with Google

Basic Information

ID Number: 2 **Priority:** HIGH **Type:** Regular

Short Description

This UC allows users to login or continue with Google if we have a Google account

Trigger

This UC starts when the user enters their email to the system.

Actors

Primary: User **Secondary:** Google

Preconditions

User must have an activated or valid Google email.

Relationships

Extends: Send Welcome Email **Includes:** N/A **Generalization/Specialization:** N/A

Major Inputs

- **Google account** (Source: User)
- **Authentication request to Google** (Source: System)

Major Outputs

- **Success or error message** (Destination: User)
- **Authentication response** (Destination: Google)

Main Flow

1. The user click continue with Google.
   Information: System displays ”continue with Google” button
2. Authentication request.
   Information: A request sent to Google for user authentication and get JWT and sent it to the system, while the system will generate a our JWT
3. Authentication response.
   Information: Confirmation of successful authentication

Alternate Flows

- User cancels authentication
- Authentication failure

Exceptions

- Invalid Google email
- Network issue

Conclusion

Post-conditions

The user is successfully authenticated and redirected to the main application interface

Business Rules

- User should have valid Google account

Special Requirements

Sign-in page should display clear button and error message

Connect WhatsApp

Basic Information

ID Number: 3 **Priority:** HIGH **Type:** Regular

Short Description

This UC allows the user to connect their WhatsApp account to our system.

Trigger

This UC is triggered when the user clicks on ”Connect WhatsApp” button in the app.

Actors

Primary: User **Secondary:** WhatsApp

Preconditions

User must be logged in

Relationships

Extends: N/A **Includes:** N/A **Generalization/Specialization:** N/A

Major Inputs

- **WhatsApp account number** (Source: User)
- **WhatsApp linking code** (Source: User)

Major Outputs

- **WhatsApp linking code** (Destination: WhatsApp)
- **WhatsApp auth creds** (Destination: System)

Main Flow

1. The user clicks ”Connect WhatsApp” button.
   Information: System asks for phone number via a dialog
2. The user enters their WhatsApp account phone number.
   Information: WhatsApp sends the linking code.
3. The user enters the linking code shown in the WhatsApp app in our app.
   Information: Confirmation of successful connection

Alternate Flows

None

Exceptions

- **Wrong linking code:** If the user enters a wrong linking code, the connection of the WhatsApp account will fail unless they enter the correct code.
- **Network issue:** A network issue interrupting the communication between the app, the server, and WhatsApp.

Conclusion

The UC ends when the user has a connected WhatsApp account.

Post-conditions

The system has access to the user's WhatsApp account.

Special Requirements

None

Continue with Google

Basic Information

ID Number: 4 **Priority:** #2 **Type:** HIGH

Regular

Short Description

This UC allows users to login or continue with Google if we have a Google account

Trigger

This UC starts when the user enters their email to the system.

Actors

Primary: User **Secondary:** Google

Preconditions

User must have an activated or valid Google email.

Relationships

Extends: Send Welcome Email **Includes:** N/A **Generalization/Specialization:** N/A

Major Inputs

- **Google account** (Source: User)
- **Authentication request to Google** (Source: System)

Major Outputs

- **Success or error message** (Destination: User)
- **Authentication response** (Destination: Google)

Main Flow

1. The user click continue with Google.
   Information: System displays ”continue with Google” button
2. Authentication request.
   Information: A request sent to Google for user authentication and get JWT and sent it to the system, while the system will generate a our JWT
3. Authentication response.
   Information: Confirmation of successful authentication

Alternate Flows

- User cancels authentication
- Authentication failure

Exceptions

- Invalid Google email
- Network issue

Conclusion

Post-conditions

The user is successfully authenticated and redirected to the main application interface

Business Rules

- User should have valid Google account

Special Requirements

Sign-in page should display clear button and error message

Set Event Priority

Basic Information

ID Number: 5 **Priority:** HIGH **Type:** Regular

Short Description

The user adds or modifies an event, and the system detects a conflict with an existing event in the calendar.

Trigger

The user adds or modifies an event, and the system detects a conflict with an existing event in the calendar.

Actors

Primary: User **Secondary:** None

Preconditions

The user is logged into the application

Relationships

Extends: Send Welcome Email **Includes:** N/A **Generalization/Specialization:** N/A

Major Inputs

- **Conflicting events (both the new event and the existing event)** (Source: User)
- **Priority decision** (Source: User)

Major Outputs

- **The system prioritizes the selected event based on the user's choice** (Destination: Calendar)
- **Confirmation messages** (Destination: User Interface)

Main Flow

1. Conflict Detection
   Information: The system detects overlapping events and prompts the user with a conflict
2. Choose Priority Option
   Information: The system displays a dialogue box with both conflicting events and asks the user to select which event should take priority.
3. Resolve Conflict
   Information: The system offers options to either reschedule the lower-priority event or keep both events with a conflict warning
4. Save
   Information: • After the user selects the priority, the system saves the decision, either reschedules the lower-priority event or marks it as conflicting and updates the calendar accordingly.

Alternate Flows

1. User clicks on a date on the calendar to bring up a popup with event fields and the date is pre-filled.
2. The user can add a different color for the events

Exceptions

- No Available Time Slots

Post-conditions

The conflicting events are resolved, and the calendar reflects the user's decision on which event to prioritize.

Special Requirements

The user interface must clearly show conflict warnings.

Conclusion

The use case ends when the user's priority decision is saved, and the conflicting events are either rescheduled or maintained with a conflict warning.

Schedule Prayer Times

Basic Information

ID Number: 6 **Priority:** HIGH **Type:** Regular

Short Description

Allows users to create, edit, and manage their prayer time schedules.

Trigger

User selects the option to schedule prayer times.

Actors

Primary: User **Secondary:** None

Preconditions

User must be logged into the system.

Relationships

Extends: N/A **Includes:** N/A **Generalization/Specialization:** N/A

Major Inputs

- **User's prayer time details** (Source: user)
- **(date, time)**

Major Outputs

- **Confirmation of scheduled prayer times, calendar entries.**

Main Flow

1. User navigates to the prayer scheduling page.
   Information: User is presented with a form to enter prayer times. Display of the scheduling interface.
2. User inputs prayer time details.
   Information: Input fields for time, date, and prayer type. User's input is captured for validation.
3. User saves the schedule.
   Information: System checks for valid input. Confirmation message displayed.
4. System confirms schedule creation.
   Information: User receives a notification. Scheduled prayer times are saved in the system.

Alternate Flows

- If input is invalid, display error messages.

Exceptions

- If there's a system error, display a relevant error message.

Post-conditions

The system generates calendar entries for the scheduled prayer times.

Special Requirements

The system must support notifications for prayer times.

Conclusion

User's prayer times are successfully scheduled.

Business Rules

- Prayer times must be within valid time ranges.

Receive Event Notifications

Basic Information

ID Number: 7 **Priority:** HIGH **Type:** Regular

Short Description

Allows users to receive notifications about upcoming events.

Trigger

User subscribes to notifications for specific events.

Actors

Primary: User **Secondary:**

Preconditions

User must be logged into the system and subscribed to event notifications.

Relationships

Extends: N/A **Includes:** User Preferences Management **Generalization/Specialization:** N/A

Major Inputs

- **User's subscription preferences,event details.** (Source: User)

Major Outputs

- **Notifications sent to users,confirmation of subscription.** (Destination: System)

Main Flow

1. Notifications sent to users, confirmation of subscription.
   Information: User is presented with options to manage notification preferences. Display of current subscription status and options.
2. User selects events to receive notifications for.
   Information: Options for different types of events prayer times, community events.User's selections are captured for processing.
3. User saves the notification preferences.
   Information: System validates user selections. Confirmation message displayed.
4. System schedules notifications based on user preferences.
   Information: Notifications are set to trigger at specified times. Notifications are queued for delivery.

Alternate Flows

If user cancels the subscription, display confirmation message.

Exceptions

- **If there's a system error, display a relevant error message.:**

Conclusion

User successfully subscribes to event notifications.

Post-conditions

Notifications are sent to users as per their preferences.

Special Requirements

The system must support multiple notification channels (email, app notifications).

Business Rules

Notifications must be sent in accordance with user preferences.

Add Event Manually

Basic Information

ID Number: 8 **Priority:** HIGH **Type:** Regular

Short Description

This UC allows users to add events manually

Trigger

The user clicks and adds the events.

Actors

Primary: User **Secondary:** None

Preconditions

The user is logged into the application

Relationships

Extends: N/Al **Includes:** N/A **Generalization/Specialization:** N/A

Major Inputs

- **Event Name** (Source: User)
- **Event Location** (Source: User)
- **Event Date (Start and End)** (Source: User)
- **Event Time (Start and End)** (Source: User)
- **Event Description** (Source: User)
- **Notifications/Reminders** (Source: User)

Major Outputs

- **The event is displayed on the calendar with its details and duration** (Destination: Calendar)

Main Flow

1. The name of the event (e.g., ”Meeting with Client”)
   Information: Event is saved and displayed on the user’s calendar
2. The start and end time of the event.
   Information: The system displays the event in the correct time slot with its duration, while checking for conflicts and scheduling notifications
3. The User optionally adds a description
   Information: The system stores and displays description of the event.
4. Set reminders for the event.
   Information: Notification or reminder set for the event

Alternate Flows

1. User clicks on a date on the calendar to bring up a popup with event fields and the date is pre-filled.
2. The user can add a different color for the events.

Exceptions

- Invalid date and time format
- The event’s start and end times conflict with an existing event
- The user attempts to save the event without filling in mandatory fields

Conclusion

The UC ends when the event has been successfully added to the calendar, and the user sees it displayed.

Post-conditions

The event is successfully added to the calendar, visible in the correct time slot, and notifications are set as per user preferences

Special Requirements

The interface must be simple and allowing users to input events with less efforts.

Bibliography

[Calendi, 2024] Calendi (2024). Calendi: Ai calendar system. Accessed: 2024-09-18.

[Clockwise, 2024] Clockwise (2024). Clockwise: Smart calendar assistant. Accessed: 2024-09-18.

[Motion, 2024] Motion (2024). Motion: Intelligent calendar. Accessed: 2024-09-18.

[Reclaim, 2024] Reclaim (2024). Reclaim ai: Intelligent time management. Accessed: 2024-09-18.

[Tungare et al., 2008] Tungare, M., Perez-Quinones, M., and Sams, A. (2008). An exploratory study of calendar use.