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

CalDAV	Calendaring Extensions to WebDAV
WebDAV	Web Distributed Authoring and Versioning
HTTPS	HyperText Transfer Protocol Secure
iOS	iPhone Operating System
N/A	Not Applicable
UC	Use Case

1 INTRODUCTION

1.1 Background of the Project

Calendars have been around a long time now, and they are a handy tool for humans. Both who are busy and who want to plan their days. People throughout history have used paper for calendars, but now with technology, things have changed. Calendars are digital now, and they can even be shared with others!

As the world is becoming one big village with globalization, people tend to squeeze every last minute of their days since competition is higher. Calendars help in that since they allow people to plan their days easily and keep track of when to meet people and do other activies.

1.2 Problem Statement

Keeping your calendar up to date with information is challenging, especially with the rise of many informal communcation channels like WhatsApp. People nowadays discuss when and where they will meet using those informal communication tools. This leads to calendars being out of sync from real life events you are committed to and might harm relations. The problem lies in the cumberness of adding events to a calendar manually, and sometimes out of busyness, you just forget that you didn't add the event to your calendar. Our Jadwal app aims to solve this issue for users in an elegant way that makes it seamless to manage your time confidently.

1.3 Objectives of the Project

The main objectives of Jadwal are:

- To develop an intelligent calendar management system that automatically extracts events from the informal communication channels like WhatsApp and adds them to the user's main calendar.
- To create a user friendly interface that allows users to easily add events to the calendar.
- To implement a 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 Project

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 Project

Jadwal's significance can be summarized in the following points:

1. **Time is Money**: Since time is the only asset you can't get more of, Jadwal tries to make it less painful and less time consuming to have a good calendar throughout your day by parsing events from your informal communication channels like WhatsApp automatically.
2. **Prayer First Calendar**: Prayer times come first, then your daily scheduled items.
3. **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.
4. **Conflict Resolution**: The smart resolution system helps users identify and resolve scheduling conflicts efficiently, reducing stress and improving overall time management.
5. **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 Project

Nothing is perfect, and our project is not an 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 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**.

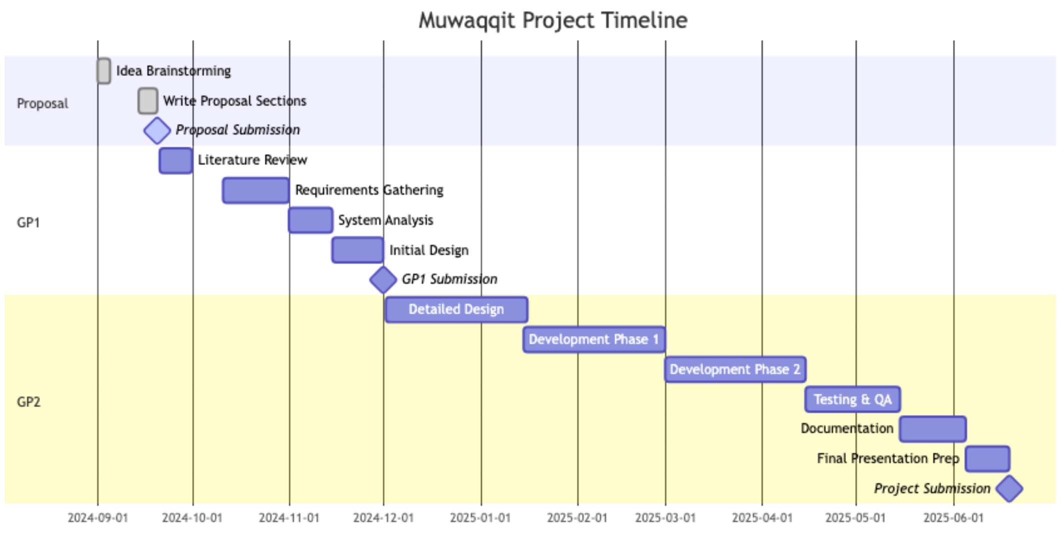
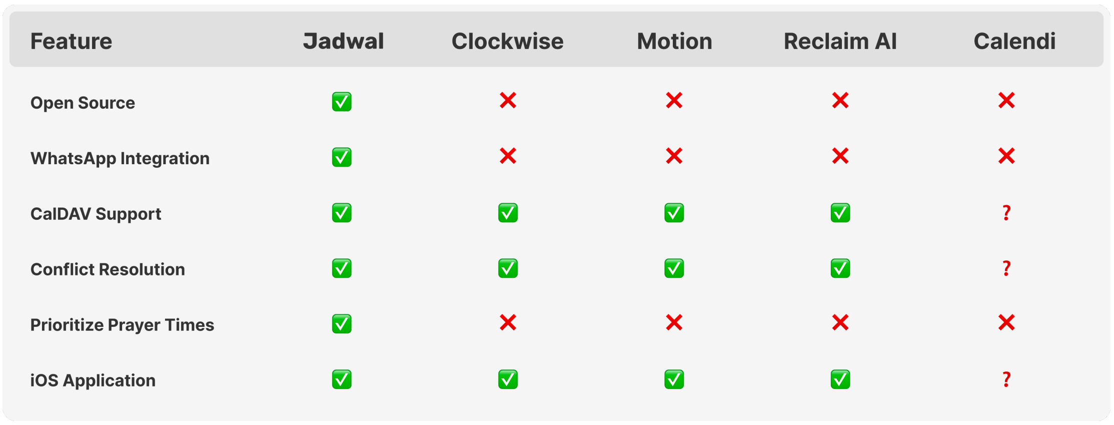


Figure 1.1: Project Gantt Chart

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.
- 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 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 16.0 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 Arabic and English languages.
- **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.

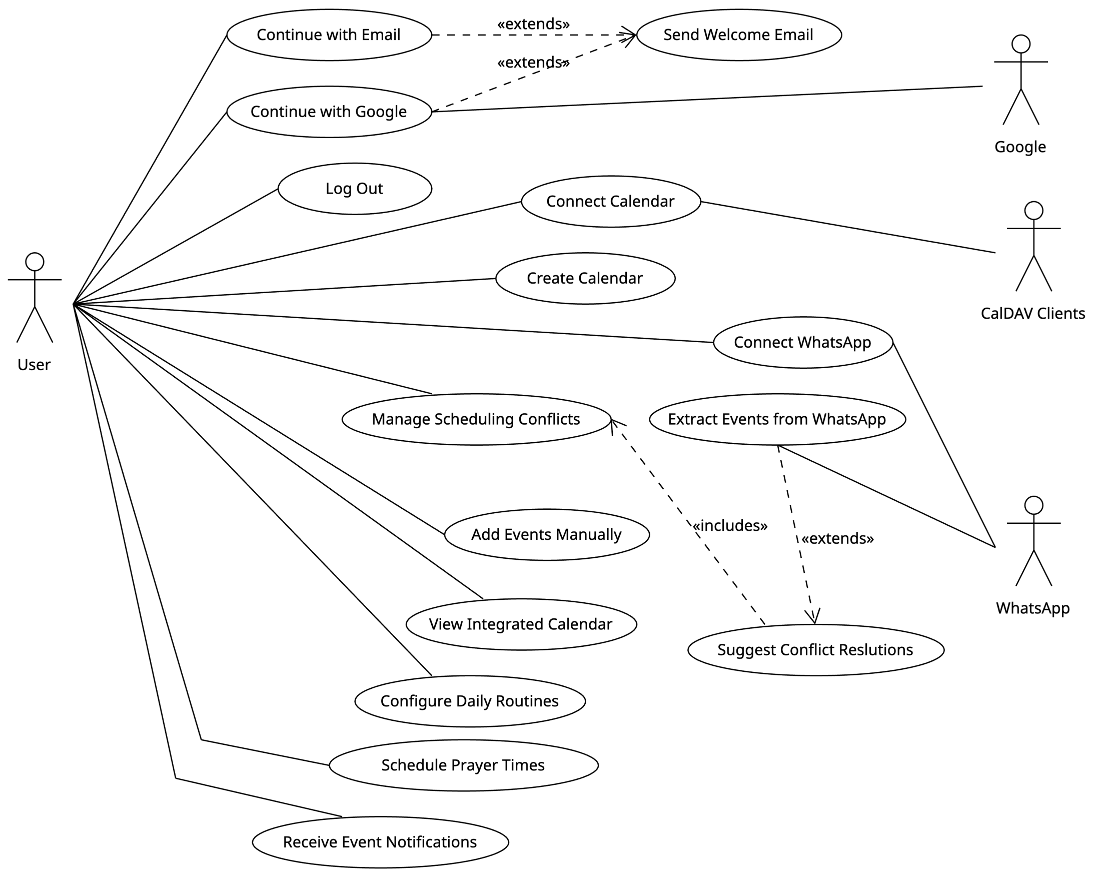


Figure 3.1: Use Case Diagram of Jadwal

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)
- **JWT** (Destination: App)

Main Flow

1. The user enters their email.
   Information: System displays an email input field.
2. System creates an account if the user has no account, and then generates and sends the magic link.
   Information: App displays “Check your email” message.
3. The user clicks the magic link in the email.
   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

-

Exceptions

- Invalid email format.
- Magic link token expired or invalid.
- **Request sending failure**: If sending the request fails due to network issues, the system prompts the user to try again.

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 magic link email.

Continue with Google

Basic Information

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

Short Description

This UC allows users to login or sign up with their Google account.

Trigger

This UC starts when the user clicks “Continue with Google” button in the app.

Actors

Primary: User **Secondary:** Google

Preconditions

The user must have an active Google account.

Relationships

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

Major Inputs

- **Google access token** (Source: User)

Major Outputs

- **Authentication response** (Destination: User)
- **JWT** (Destination: App)

Main Flow

1. The user click continue with Google.
   Information: App uses OAuth to authenticate with Google
2. App sends Google access token to the system.
   Information: System verifies the token is issued for us and then issues JWT for usage within the app.

Alternate Flows

- The user cancels the authentication request.

Exceptions

- Google access token invalid or expired.
- **Request sending failure**: If sending the request fails due to network issues, the system prompts the user to try again.

Conclusion

This UC ends when the user is logged in.

Post-conditions

The system generates a JWT.

Special Requirements

A google client must be present for the validation of the access token to be possible.

Send Welcome Email

Basic Information

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

Short Description

This UC welcomes the user to the platform.

Trigger

This UC starts when the user account is created.

Actors

Primary: User **Secondary:** None

Preconditions

User account must be created in the system.

Relationships

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

Major Inputs

- **User name** (Source: System)
- **Welcome email template** (Source: System)

Major Outputs

- **Welcome email** (Destination: User)

Main Flow

1. The system fetches the user information.
   Information: The database is used.
2. The system fetches the send welcome email template.
   Information: The template is filled with the user name.
3. The system sends the email with the template.
   Information: The email is received by the user welcoming them.

Exceptions

- Email server is down.

Conclusion

This UC ends when the user receives an email from us welcoming them.

Special Requirements

An email server must be present to send welcome email.

Logout

Basic Information

ID Number: 4 **Priority:** High **Type:** Regular

Short Description

This UC allows the user to logout from the app.

Trigger

This UC is triggered when logout button in the settings page is clicked.

Actors

Primary: User **Secondary:**

Preconditions

The user must be logged in.

Relationships

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

Main Flow

1. The user click the logout button
   Information: The app deletes the JWT and moves the user to the onboarding screen.

Conclusion

The user is logged out.

Post-conditions

The user will be logged out from the system

Connect Calendar

Basic Information

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

Short Description

This UC allows the user to connect their external calendars to our system.

Trigger

This UC is triggered when the user selects the option to connect an external calendar in the app.

Actors

Primary: User **Secondary:** CalDAV

Preconditions

User must be logged in

Relationships

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

Major Inputs

- **CalDAV login credentials and Name** (Source: User)

Major Outputs

- **Calendar data sync status** (Destination: System)

Main Flow

1. The user clicks the "Connect Calendar" option in the app.
   Information: The app asks the user to enter their CalDAV credentials along with a user provided name.
2. The system talks to the external calendar system via the credentials provided by the user.
   Information: The system adds the received calendar data to the database and a connection success status is shown to the user.
3. The system saves the CalDAV credentials securely in the database.
   Information: The credentials are encrypted before storing them.

Alternate Flows

1. If the credentials are wrong or the request times out, the user can retry the request again.

Exceptions

- Invalid credentials.
- Network issue with CalDAV server.

Conclusion

The UC ends when the user has a successfully connected and synced external calendar with our system.

Post-conditions

The system has access to the user's external calendar, and events are synced and displayed for the user in the app..

Special Requirements

The system must handle multiple calendars efficiently.

Create Calendar

Basic Information

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

Short Description

This UC allows the user to create a calendar in our system.

Trigger

This UC is triggered when the user clicks “Create Calendar” in the app.

Actors

Primary: User **Secondary:** None

Preconditions

User must be logged in

Relationships

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

Major Inputs

- **Calendar name** (Source: User)
- **Calendar color** (Source: User)

Major Outputs

- **Calendar** (Destination: System)
- **Calendar creation status** (Destination: App)

Main Flow

1. The user clicks the “Create Calendar” button in the app.
   Information: The app asks the user to enter the calendar name and choose a calendar color.
2. The user submits the form for calendar information.
   Information: The system creates a calendar for the user in the system.

Alternate Flows

- If the requets of creating a calendar fails, prompt the user to try again.

Exceptions

- **Calendar creation failure:** If creation of a calendar fails due to network issues, the system prompts the user to try again.

Conclusion

The UC ends when the user has a new calendar created successfully.

Post-conditions

The user has a new calendar in the list of calendars in the app.

Connect WhatsApp

Basic Information

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

Short Description

This UC allows the user to connect their WhatsApp account to the 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 phone number** (Source: User)
- **WhatsApp linking code** (Source: User)

Major Outputs

- **WhatsApp auth credentials** (Destination: System)

Main Flow

1. The user clicks “Connect WhatsApp” button.
   Information: The system asks for the user’s WhatsApp phone number.
2. The user enters their WhatsApp phone number.
   Information: WhatsApp shows the linking code in their app.
3. The user enters the linking code in our app.
   Information: The app shows a success screen if connection was sucessful.

Alternate Flows

- If the WhatsApp connection fails, the user must redo the steps and try again.
- If the user enters a wrong linking code, the connection of the WhatsApp account will fail unless they enter the correct code.

Exceptions

- **Wrong linking code:** If the user enters a wrong linking code too many times, the connection of the WhatsApp account will fail.
- **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 in the system.

Post-conditions

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

Extract Events from WhatsApp

Basic Information

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

Short Description

System monitors WhatsApp messages of connected WhatsApp accounts and extract event details adding them to the user's calendar.

Trigger

A messages is sent to the connected WhatsApp account of an arbitrary user in our system.

Actors

Primary: WhatsApp **Secondary:**

Preconditions

At least one WhatsApp account must be connected.

Relationships

Extends: Suggest Conflict Resolution **Includes:** N/A
Generalization/Specialization: N/A

Major Inputs

- **Messages sent to currently selected user** (Source: WhatsApp)

Major Outputs

- **Extracted event details** (Destination: System)

Main Flow

1. A sent message is received and the user has replied and 30 seconds have passed without any interruptions.
   Information: System reads a few messages before the current one to have context.
2. Add the detected event to the user's calendar.
   Information: Send a notification to the user telling them about the newly added event.

Alternate Flows

- If there is a conflict adding this event, send a notification telling the user that there is a conflict they need to resolve.

Exceptions

- If there is an error during extraction of events, fail silently and log it to a specific table in the database for debugging later by developers.

Conclusion

System successfully adds the event to the user's calendar and notifies the user of the added event.

Post-conditions

The event is added to the user's calendar.

Business Rules

- Only messages with events and its surrounding context shall be analyzed.
- System must wait for user's reply before analyzing the messages.
- System must wait for 30 seconds before initiating the analysis on messages after the user replies and reset as long the conversation is onggoing.

Special Requirements

The system must have access to the user's WhatsApp account as a client to receive and read messages.

Suggest Conflict Resolutions

Basic Information

ID Number: 9 **Priority:** High **Type:** Regular

Short Description

This UC gives the user all the conflicts and possible ways to resolve it.

Trigger

The UC is triggered when a conflict is detected between any overlapping event

Actors

Primary: User **Secondary:** Calendar System, WhatsApp

Preconditions

The calendar must have events

Relationships

Extends: None **Includes:** Manage Scheduling Conflicts
Generalization/Specialization: None

Major Inputs

- Overlapping events (Source: User or WhatsApp extraction)
- User suggested resolution (Source: User)

Major Outputs

- Resolution options (Destination: User Interface)
- Updated calendar schedule (Destination: Calendar)

Main Flow

1. Conflict Detection

   Information: The system detects overlapping of events either added manually or extracted from WhatsApp.
2. Offer Resolutions

   Information: The system provides the user with the list of resolution options.
   - By moving the overlapping event to another time slot.
   - Keep both events with a conflict warning.
3. User Selection

   Information: The user moves the overlapping event to another time slot by the list provided by the system
4. Implement Resolution

   Information: The system makes the changes in the calendar respectively

Alternate Flows

1. N/A

Exceptions

- If the system doesn't get any possible way to resolve the conflict then the system would mark both the event as conflicting or displays to delete one of the event.
- If the user doesn't choose any option to resolve the conflict then both the events are makrd as conflicting.

Conclusion

The UC ends when the user gives it's input for resolution weather if it's to reschedule the event, delete it or leave it without resolving and it is reflected in on the calendar.

Post-conditions

The conflicting events in the calendar are either resolved or marked as conflicting

Special Requirements

The system give feasible conflict resolution options

Manage Scheduling Conflicts

Basic Information

ID Number: 10 **Priority:** High **Type:** Regular

Short Description

This UC allows the user to manage scheduling conflicts by suggesting resolutions when overlapping events are detected, including events extracted from WhatsApp.

Trigger

This UC is triggered when the user adds, modifies, or extracts an event (e.g., from WhatsApp) that overlaps with an existing event.

Actors

Primary: User **Secondary:** Calendar System, WhatsApp

Preconditions

The user is logged into the application, and WhatsApp events are synchronized with the calendar.

Relationships

Extends: Includes: Suggest Conflict Resolutions **Includes:** Extends: Extract Events from WhatsApp, Set Event Priorities
Generalization/Specialization: None

Major Inputs

- **Conflicting events** (Source: User or WhatsApp extraction)
- **Priority decision** (Source: User)

Major Outputs

- **Conflict resolution suggestion** (Destination: User Interface)
- **Updated event schedules** (Destination: Calendar)

Main Flow

1. The user adds, modifies, or extracts an event from WhatsApp.
2. The system detects a conflict with an existing event.
   Information: The system highlights both the new/extracted event and the existing conflicting event.
3. The system invokes Suggest Conflict Resolutions UC to offer solutions such as rescheduling or prioritizing.
4. The user selects a resolution option from the suggestions.
5. The system saves the updated schedule or marks the events with a conflict warning.

Alternate Flows

1. If the user manually chooses to prioritize one event, the Set Event Priorities UC is triggered.

Exceptions

- **No resolution selected:** If the user doesn't choose a resolution, both events remain scheduled but marked as conflicting.
- **No available time slots:** If the system cannot find suitable time slots for rescheduling.

Conclusion

The UC ends when the conflicting events are either resolved or marked as conflicting, based on the user's choice.

Post-conditions

The calendar reflects the user's decision regarding event conflicts, including those extracted from WhatsApp.

Special Requirements

The system should provide intuitive suggestions for resolving conflicts, including events extracted from WhatsApp.

Add Event Manually

Basic Information

ID Number: 11 **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.

View Integrated Calendar

Basic Information

ID Number: 12 **Priority:** High **Type:** Regular

Short Description

Allows users to view their integrated calendar with all scheduled events and reminders.

Trigger

User selects the option to view the integrated calendar.

Actors

Primary: User **Secondary:**

Preconditions

User must be logged into the system.

Relationships

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

Major Inputs

- **User's request to view the calendar, date range.** (Source: User)

Major Outputs

- **Display of the integrated calendar with events and reminders.** (Destination: System)

Main Flow

1. Display of the integrated calendar with events and reminders.
   Information: Display of the current month with events highlighted.
2. Options to navigate through months or choose a specific date.
   Information: Calendar updates to show selected date events.
3. Events are pulled from the user's account and any linked calendars.
   Information: Display of events for the selected date.
4. Tooltips or pop-up modals show event details.
   Information: Detailed view of selected event information.

Alternate Flows

Notify the user about the absence of events. Message indicating no events scheduled.

Exceptions

- **Log error for troubleshooting.User-friendly error message displayed.:**

Conclusion

User successfully views their integrated calendar with all scheduled events.

Post-conditions

The calendar remains updated with any new events or changes.

Special Requirements

The system must support integration with external calendars (Google Calendar).

Business Rules

Events must be displayed according to user preferences (color coding, categories).

Configure Daily Routine

Basic Information

ID Number: 13 **Priority:** High **Type:** Regular

Short Description

The use case starts when the user initiates the configuration or editing of their daily routine in the app.

Trigger

The user selects the“Daily Routine” feature to configure repetitive tasks like waking up, exercise, meals, and sleeping time.

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

- **Routine tasks (e.g., wake up, jog, meals, bath time, sleep)** (Source: User)
- **Start and end times for each task.** (Source: User)
- **Recurrence pattern (daily, weekdays, weekends, etc)** (Source: User)
- **Notifications or reminders (e.g., alarms for specific tasks).** (Source: User)

Major Outputs

- **The daily routine is added to the calendar with all configured tasks displayed as recurring events.** (Destination: Calendar)

Main Flow

1. The user adds Daily tasks (wake up, meals, sleep, etc.)
   Information: Routine tasks are added to the 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

Alternate Flows

1. The user clicks on the calendar, the system displays the daily routine tasks.

Exceptions

- Conflict with tasks

Conclusion

Configuring a daily routine enables users to block out dedicated time for recurring tasks.

Post-conditions

The daily routine is successfully configured and integrated into the calendar, with appropriate visual indicators and reminders.

Special Requirements

The interface must be simple and allowing users to have the daily routines Throughout.

Schedule Prayer Times

Basic Information

ID Number: 14 **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: 15 **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.

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.