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: 30 Oct 2024
Supervised By: Dr. Inayatullah

First Semester 2024–2025

ABSTRACT

In response to the rapid globalization of modernization, time management has been a real challenge. This paper introduces Jadwal, that proposes an automatic schedule managing system where an individual can integrate multiple calendars and extract the events that have been discussed informal communication channels,such as Whatsapp

Key Objectives:

- 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.

The system will be a mobile application, designed for iOS devices.

ACKNOWLEDGMENT

First and foremost, we would like to thank our Graduation project supervisor Dr. Inayathullah for his invaluable guidance and precious time. His advise was instrumental in shaping our project and by providing support in challenging times.Moreove,he emphasized that all member should involve in each phase of the project and gain core knowledge and contribute efficiently . His stimulus and help to complete the project successful.

Secondly, we would like to thank all the professors and friends for their insightful recommendations which helped us to enhance our project.

Lastly, we would like to thank our parents for their unwavering support and for preparing us mentally and physically throughout this journey.

Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS AND TERMINOLOGY

Term	Definition
Calendar	A list of events and dates within a year that are important to an organization or to the people involved in a particular activity. [Cambridge, 2024]
Jadwal	Jadwal is a comprehensive time management tool designed to aggregate and optimize your existing calendars and data sources.
WhatsApp	WhatsApp is an alternative to SMS and offers simple, secure, reliable messaging and calling, available on phones all over the world. [WhatsApp, 2024]
CalDAV	Calendaring Extensions to WebDAV
Golang	Go is an open source project developed by a team at Google and many contributors from the open source community. [Golang, 2024]
JWT	JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties. [JWT, 2024]
WebDAV	Web Distributed Authoring and Versioning
HTTPS	HyperText Transfer Protocol Secure
iOS	iPhone Operating System
N/A	Not Applicable
UC	Use Case
ER diagram	Entity Relationship ER Diagram is a type of flowchart that illustrates how entities
Sequence Diagram	An interaction diagram that details how operations are carried out.
APIs	Application programming interface.
APNs	Apple Push Notification service

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.

People these days use multiple calendars and sometimes forget to insert an event to the correct calendar.

1.2 Problem Statement

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

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 difficult to prove that privacy risk associated with our integration isn't compromised.
- 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.
- We may face a challenges to test the app due to lack of users for testing our app

1.7 Organization of the Senior Project

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

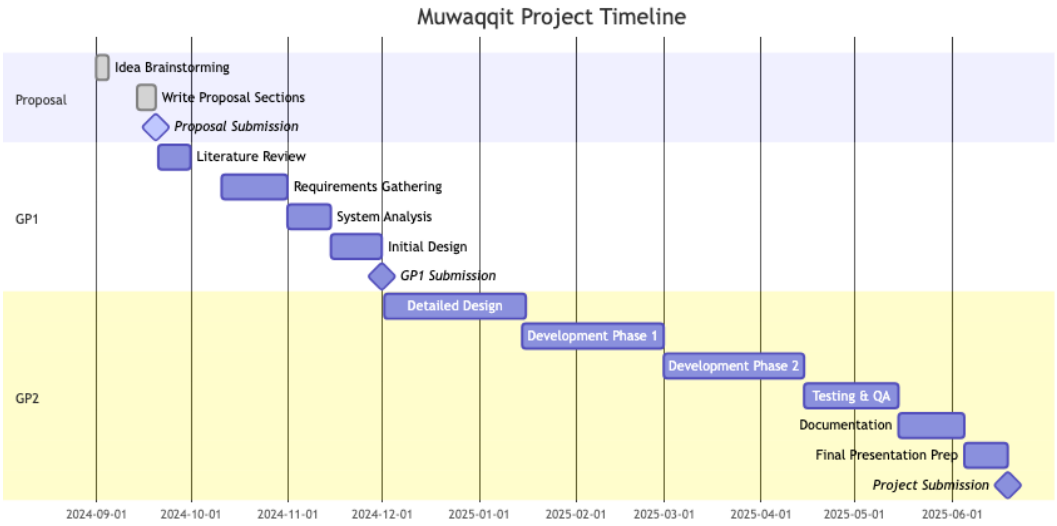

Figure 1.1: Project Gantt Chart

1.8 Conclusion

In conclusion, the shift from traditional calendars to digital ones has changed how people organize their schedule. While digital calendars offer easy and conflict free scheduling which makes managing easy. Discussion on informal communication channels like WhatsApp has impacted the planning events and commitments. This impact often leads to missed appointments and scheduling conflicts, especially for someone who is busy.

Jadwal aims to solve this issue by providing a seamless and automated solution for integrating informal communication with the application so that no one misses an event and have conflicts.

2 LITERATURE REVIEW

2.1 Introduction

In the modern world, managing time effectively is important for balancing personal and professional responsibilities. The increasing complexity of schedules usually results in missing an important event and overlapping commitments. To solve these challenges, we introduce Jadwal, an innovative iOS-based schedule management application, aims to help individuals and professionals to organize their daily lives. Jadwal focuses on solving the issue by drawing inspiration from existing solutions like Clockwise, Motion, Reclaim AI, and Calendi, Jadwal outstands by introducing advance features such as conflict resolution, task prioritization, event extraction from informal communicating channels and prioritizing prayer times, which is a unique feature which is not found in any competing applications.

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

In Figure 2.1 The table compares *Jadwal*, our project, with four other scheduling applications—Clockwise, Motion, Reclaim AI, and Calendi—based based on several key features:

- **Open Source**: *Jadwal* stands out as the only application that is open-source, allowing users and developers to access, modify, and improve the code. The other applications do not provide this.
- **WhatsApp Integration**: *Jadwal* stands out again as the only application that connects and extract the event discussed over the platform by the user.
- **CalDAV Support**: *Jadwal*, Clockwise, and Reclaim AI offer CalDAV support, which allows users to integrate calendars from different sources, while Motion and Calendi either does not sipport this.
- **Conflict Resolution**: All listed applications, including *Jadwal*, supports conflict resolution, allowing users to manage overlapping events efficiently.
- **Prioritize Prayer Times**: *Jadwal* is the only one offering this feature where the user be able to have prayer time scheduled
- **iOS Application**: *Jadwal* is available on iOS, along with Clockwise, Reclaim AI, and Motion. The availability of Calendi on iOS is uncertain.

2.2 Comparing Competitors

Below we will compare competitors more in depth, before we only talked about them in general and shown a features table.

2.2.1 Clockwise

for clockwise

2.2.2 Motion

for motion

2.2.3 Reclaim AI

for reclaim ai

2.2.4 Calendi

for calendi

2.3 Conclusion

Jadwal represents itself by offering a seamless calendar application which is user friendly and outstands by providing solution to all the problems which is not yet addressed by other tools like Clockwise, Motion, Reclaim AI, and Calendi. WhatsApp integration is Jadwal's unique features for event extraction. These features distinguish Jadwal from existing applications, Jadwal addresses to the issues that is not yet solved.

The comparison highlights Jadwal's abilities that integrate multiple calendars, resolve scheduling conflicts.

3 SYSTEM ANALYSIS AND DESIGN

3.1 Introduction

A well-designed system requires a good understanding of both functional and non-functional requirements to meet user expectations and deliver a seamless experience. Jadwal, an iOS-based intelligent scheduling application.

This section shows the functional and non-functional requirements which is the backbone of Jadwal's development. The functional requirements focus on core features, such as user authentication, calendar integration and event management. Ensuring the users can effectively manage their schedules. The non-functional requirements focuses on performance, security, compatibility, and user experience, ensuring the application stands well with the industry standards by providing efficient interface.

Combining all these requirements helps in the design and implementation of Jadwal which will lead to a better application solving real issues and meeting the needs of the user.

3.2 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 create a calendar.
- 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 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.

3.3 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.

3.4 System use-cases

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

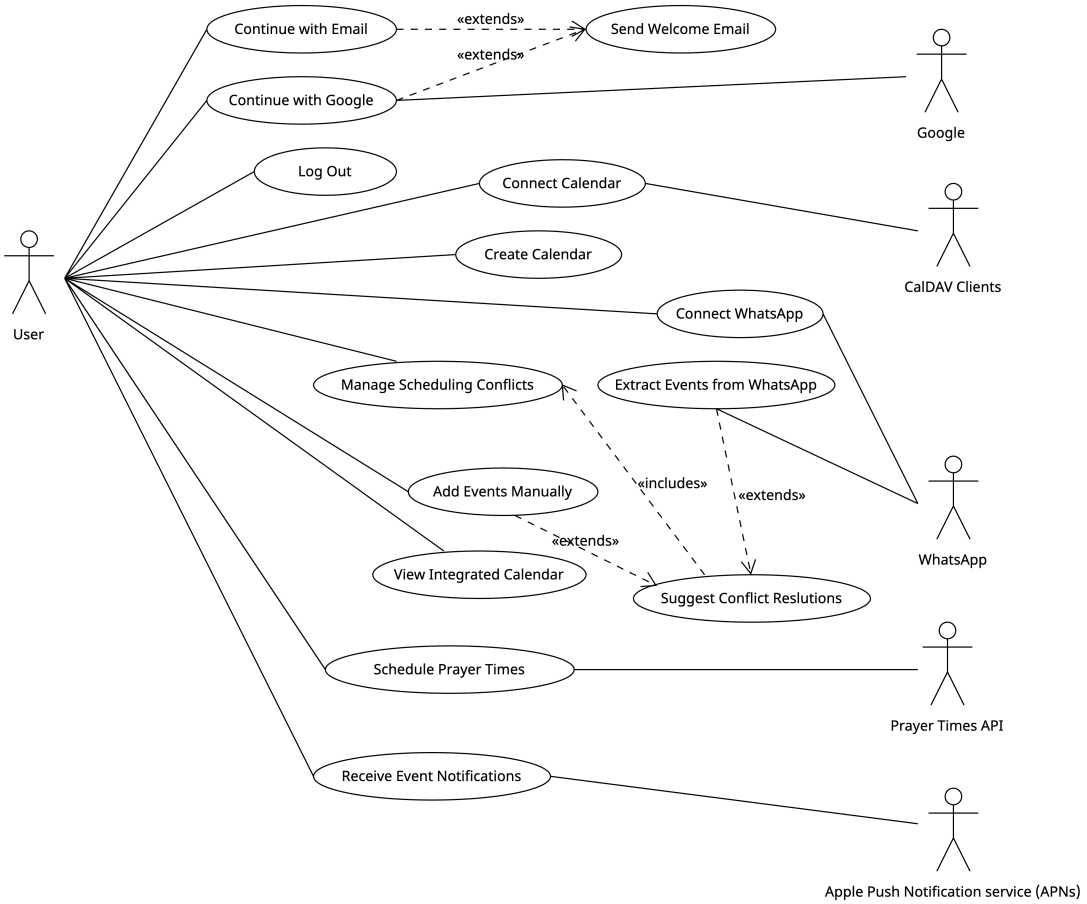

Figure 3.1: Use Case Diagram of Jadwal

Use Case 1: 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.

Table 3.1: Continue with Email

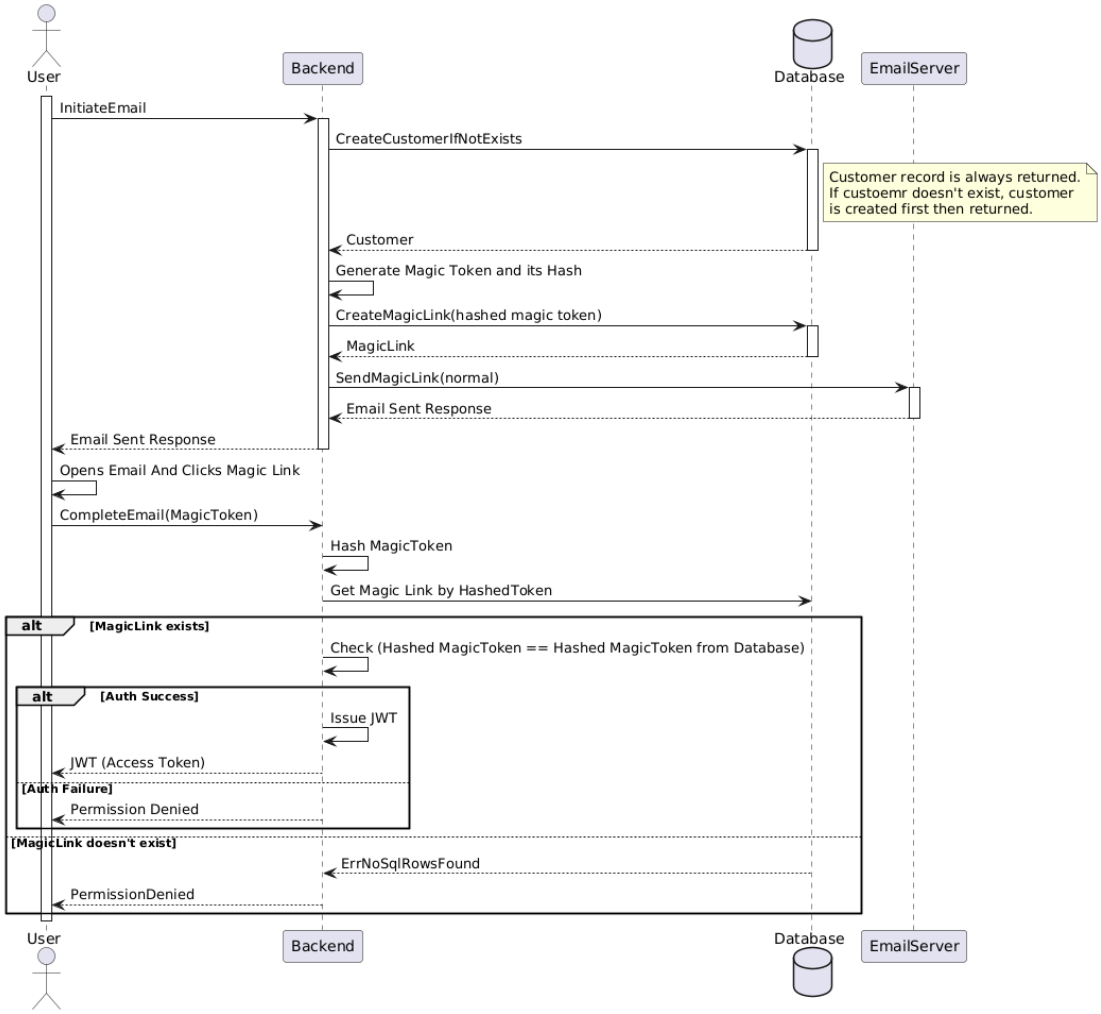

Figure 3.2: Continue with Email Sequence Diagram

Use Case 2: 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.

Table 3.2: Continue with Google

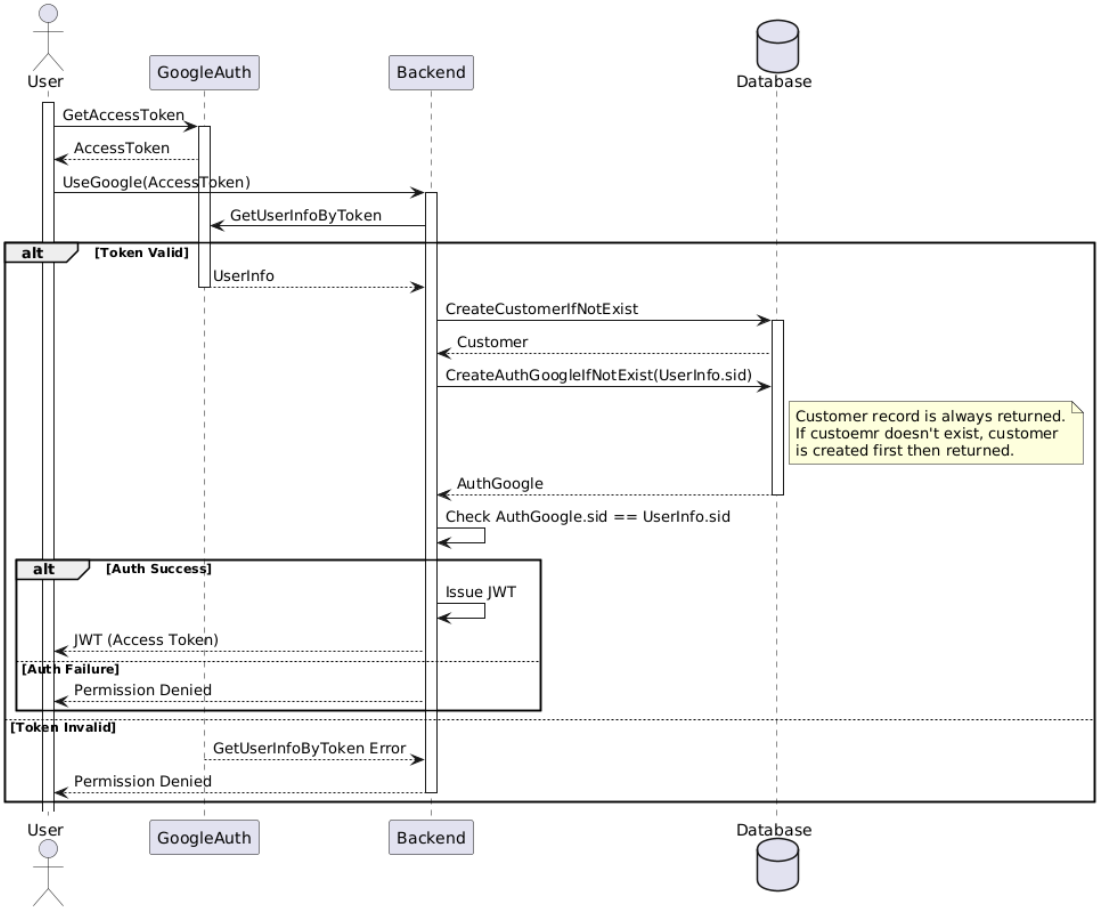

Figure 3.3: Continue with Google Sequence Diagram

Use Case 3: 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.

Table 3.3: Send Welcome Email

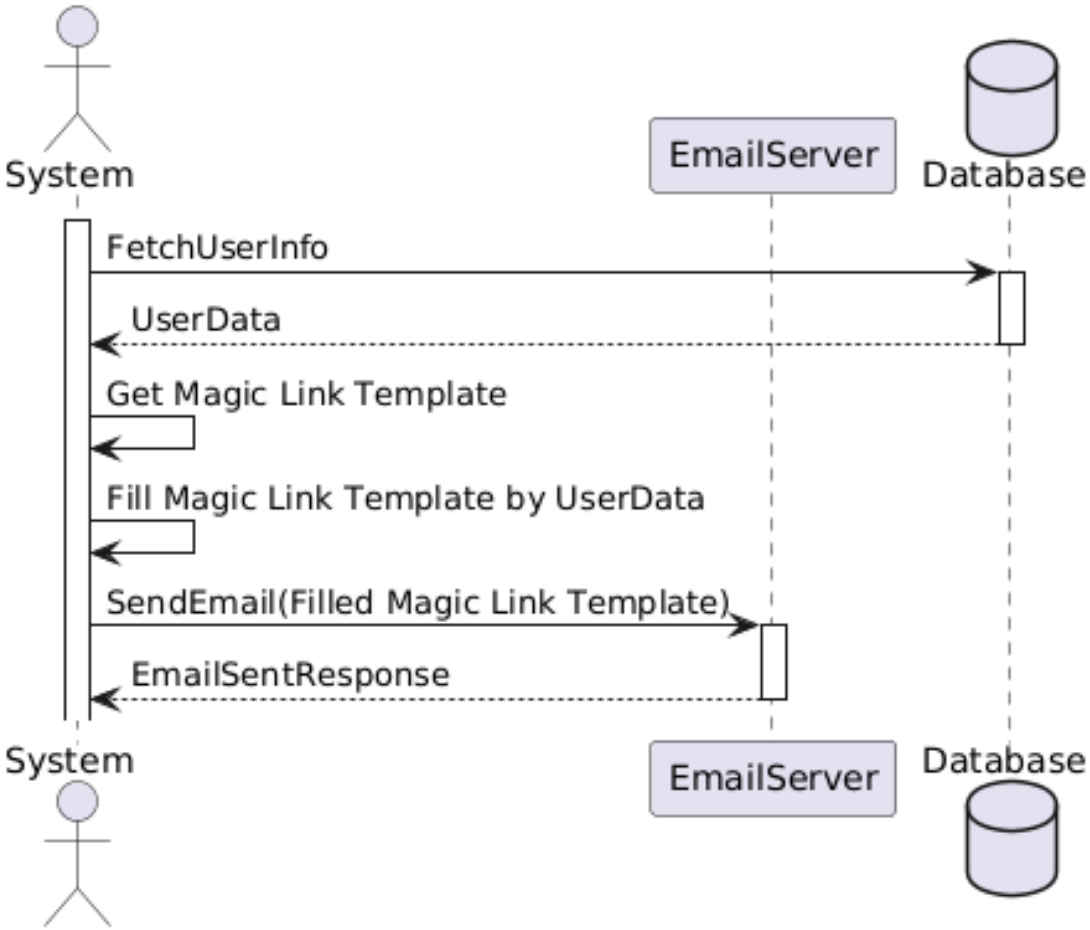

Figure 3.4: Send Welcome Email Sequence Diagram

Use Case 4: 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:** None

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

Table 3.4: Logout

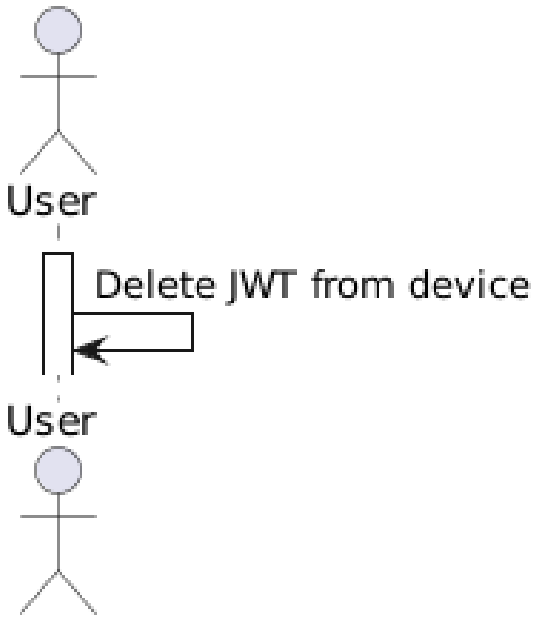

Figure 3.5: Logout Sequence Diagram

Use Case 5: 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.

Table 3.5: Connect Calendar

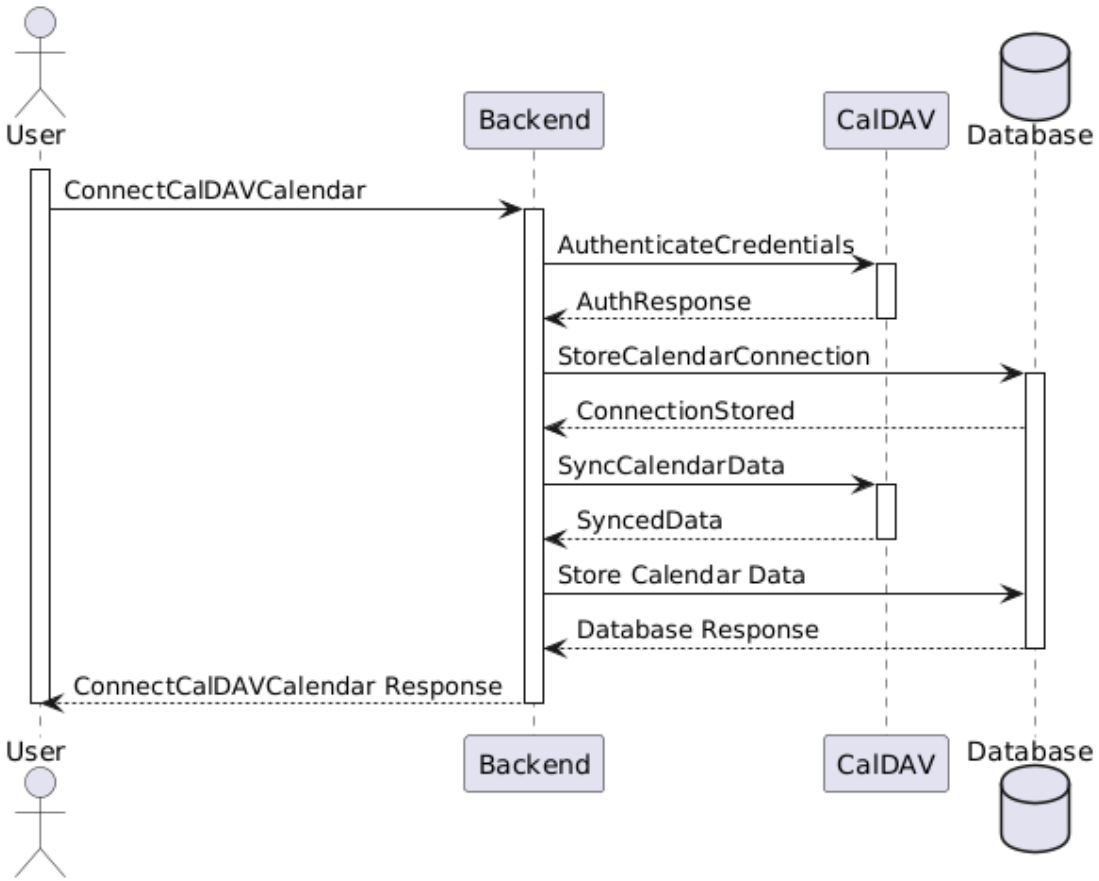

Figure 3.6: Connect Calendar Sequence Diagram

Use Case 6: 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.

Table 3.6: Create Calendar

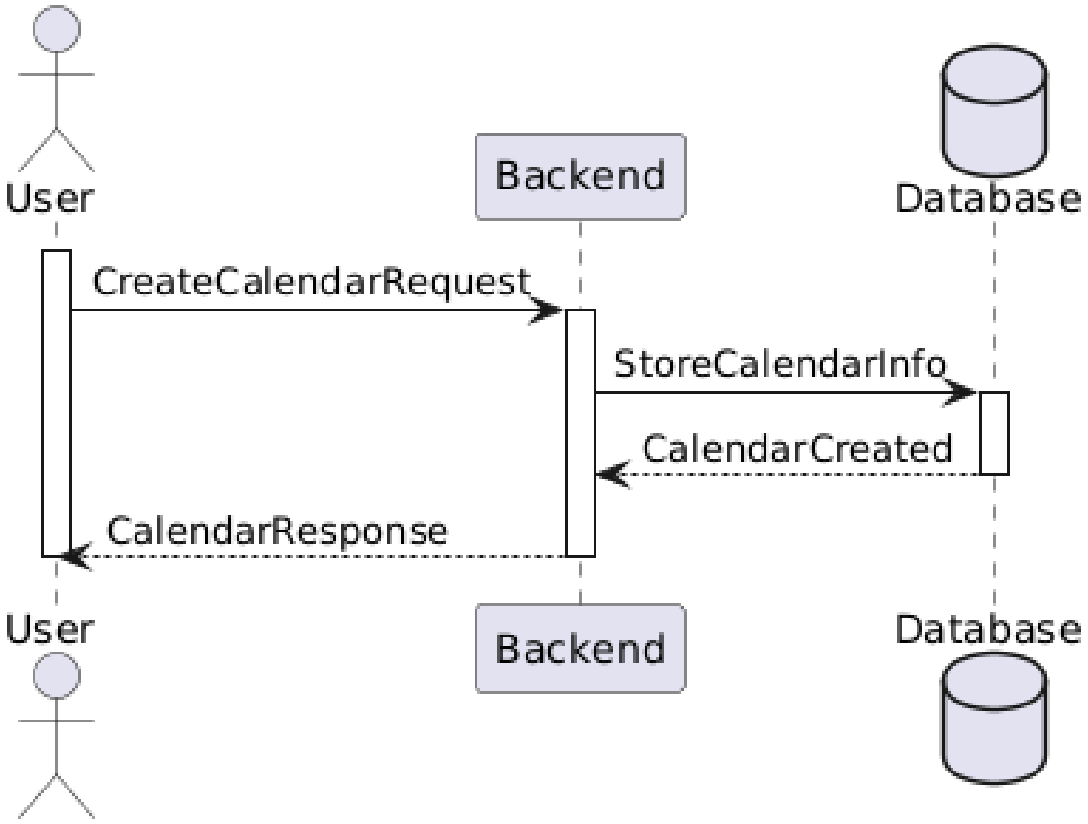

Figure 3.7: Create Calendar Sequence Diagram

Use Case 7: 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.

Table 3.7: Connect WhatsApp

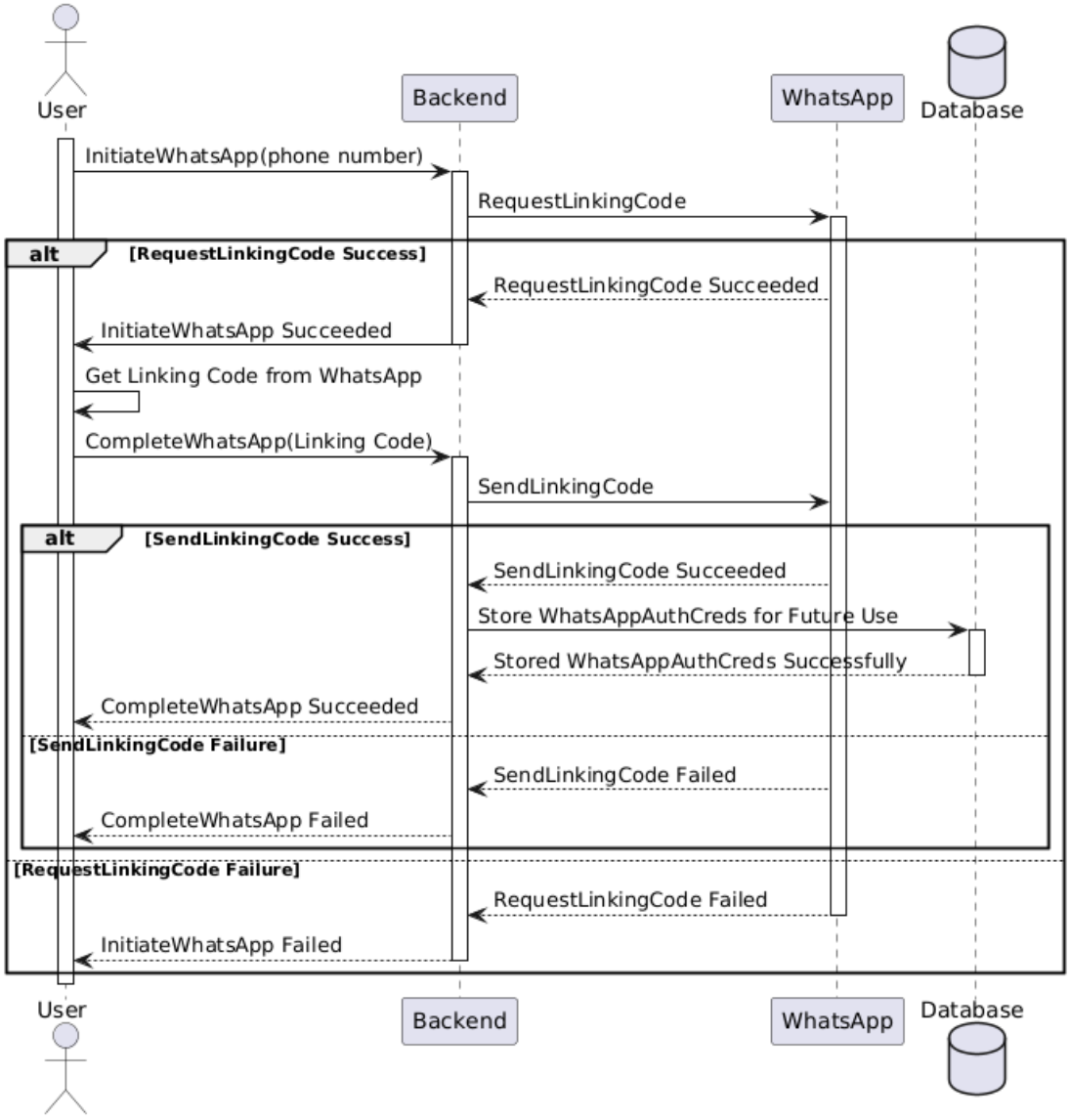

Figure 3.8: Connect WhatsApp Sequence Diagram

Use Case 8: 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.

Table 3.8: Extract Events from WhatsApp

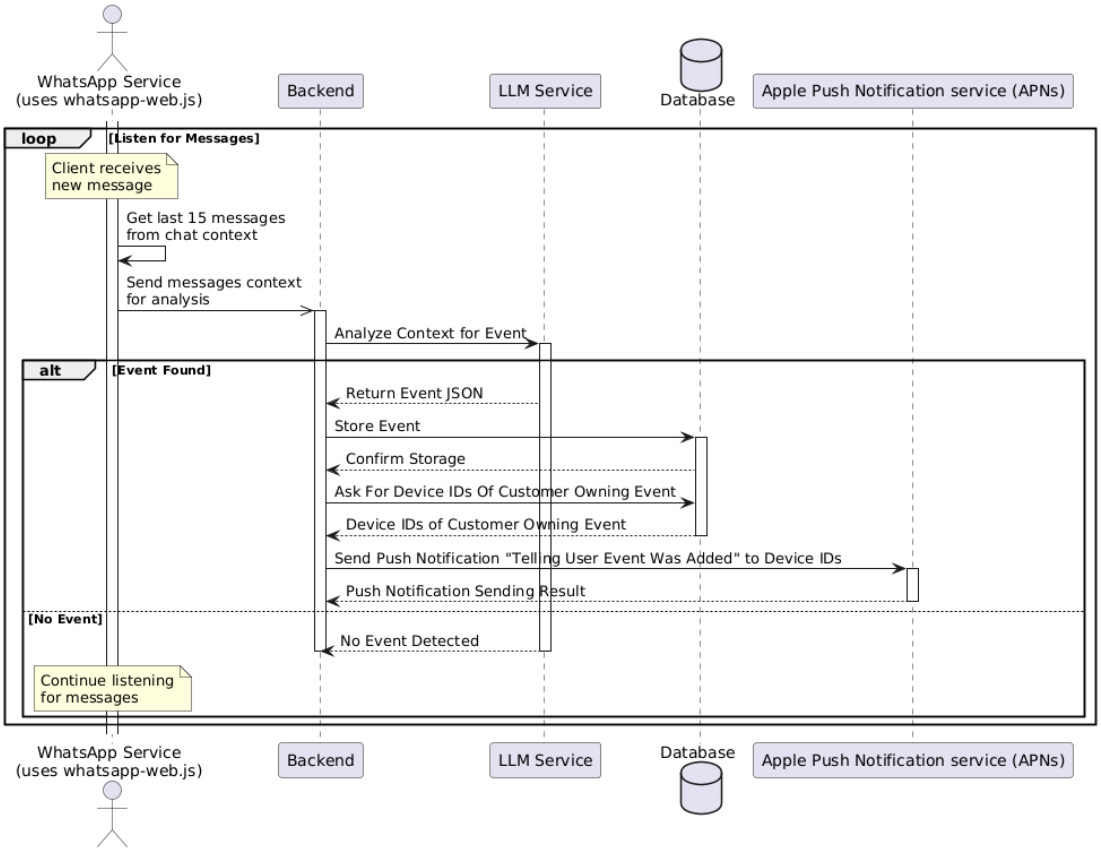

Figure 3.9: Extract Events from WhatsApp Sequence Diagram

Use Case 9: 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:** WhatsApp

Preconditions

The calendar must have events

Relationships

Extends: N/A **Includes:** Manage Scheduling Conflicts

Generalization/Specialization: N/A

Major Inputs

- **Overlapping events** (Source: Saved Events in Database)
- **User suggested resolution** (Source: User)

Major Outputs

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

Main Flow

1. The system detection conflicts

   Information: The system detects overlapping of events either added manually or extracted from WhatsApp and gives a notification to the user.

2. The system gives the suggestions for Conflicts.

   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.

Alternate Flows

1. No conflicts detected

Exceptions

- If the system doesn't get any possible way to resolve the conflict then the system would mark both the event as conflicting.

Conclusion

The UC ends when the user chooses resolution weather if to reschedule the event or leaving 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

Table 3.9: Suggest Conflict Resolutions

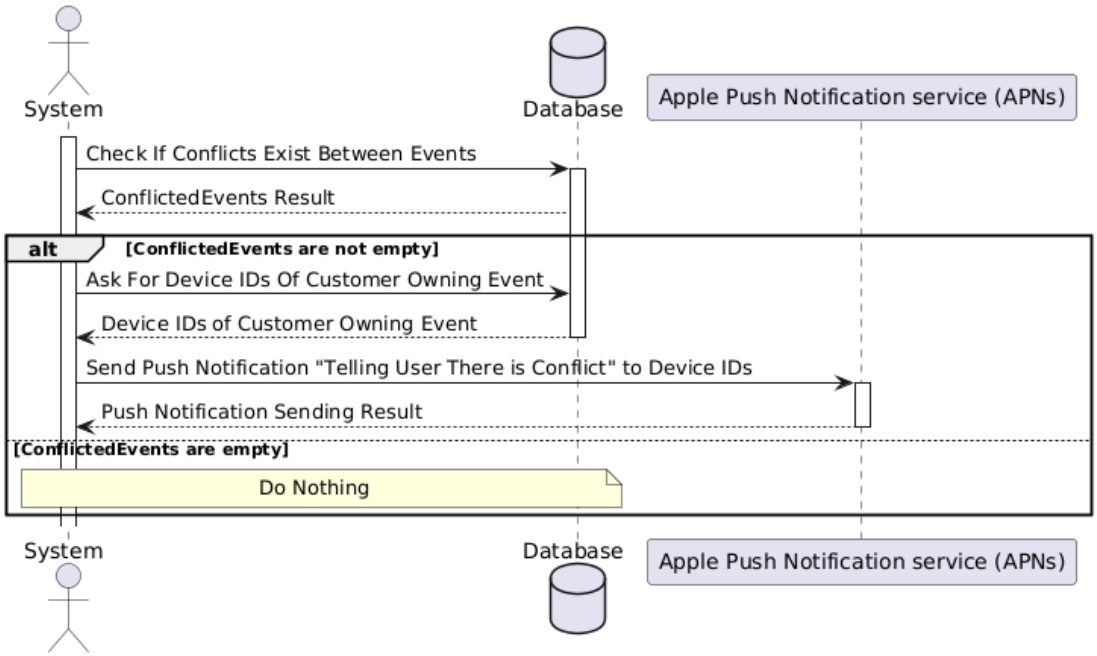

Figure 3.10: Suggest Conflict Resolutions Sequence Diagram

Use Case 10: Manage Scheduling Conflicts

Basic Information

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

Short Description

This UC allows the user to manage scheduling conflicts by suggesting resolutions when overlapping events are detected.

Trigger

This UC is triggered when an automatically added event overlaps with an existing event.

Actors

Primary: User **Secondary:** None

Preconditions

- User is logged in.
- Conflicting events list is not empty.

Relationships

Extends: N/A **Includes:** N/A

Generalization/Specialization: N/A

Major Inputs

- **Conflicting events** (Source: System)
- **Event to override** (Source: User)

Major Outputs

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

Main Flow

1. The user opens the application and clicks on the view conflicts icon

   Information: The application shows all the conflics with their resolution options

2. The user chooses the best fit option to manage each conflict

   Information: The conflict is resolved and is removed from the conflict list

Alternate Flows

1. If the user doesn't choose any option, it shows conflicting status until the user chooses any option or the event expires.
2. If the user clicks on reject the event is left overlapping.

Exceptions

- Netwok failure

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.

Special Requirements

The system should provide best suggestions for resolving conflicts.

Table 3.10: Manage Scheduling Conflicts

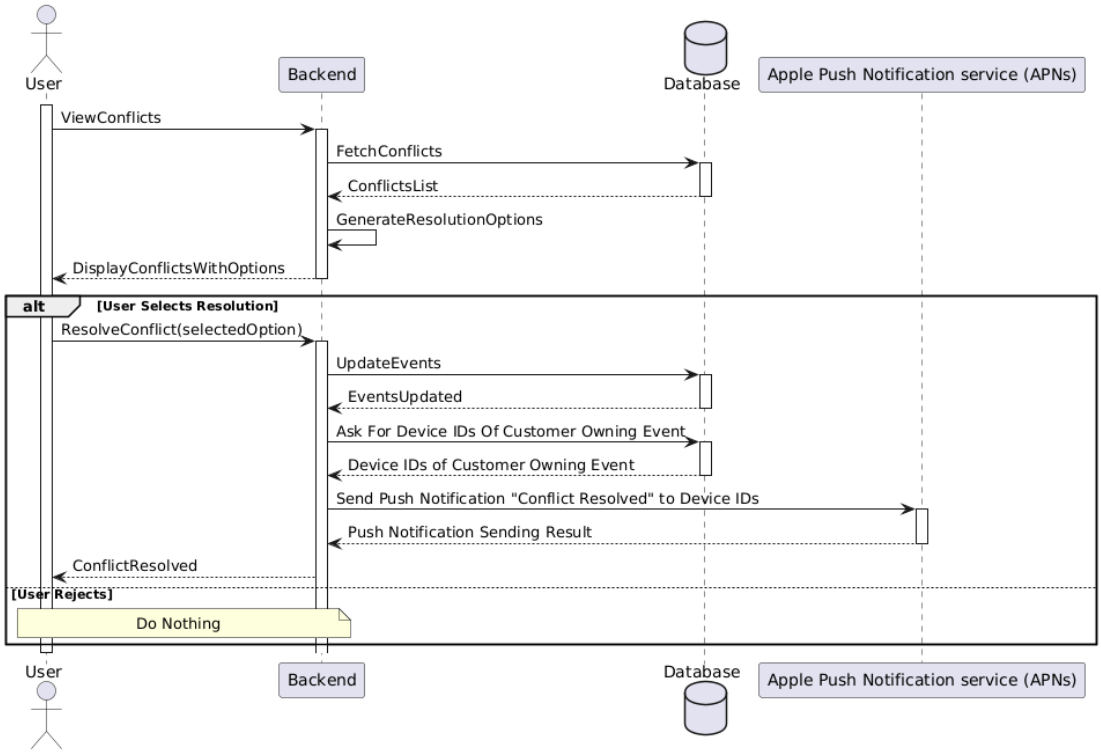

Figure 3.11: Manage Scheduling Conflicts Sequence Diagram

Use Case 11: 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 add event manually icon or a date on the calendar and adds the events.

Actors

Primary: User **Secondary:** None

Preconditions

The user is logged into the application.

Relationships

Extends: Suggest Conflict Resolutions **Includes:** N/A

Generalization/Specialization: N/A

Major Inputs

- **Event Name** (Source: User)
- **Event Location** (Source: User)
- **Is all day?** (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

- **New Calendar event** (Destination: Calendar)

Main Flow

1. The user clicks the add event manually icon or a date on the calendar.

   Information: The add event manually form is displayed.

2. The user sets the details of the event in the respective fields and saves the event.

   Information: The event is displayed on the calendar with its details.

Alternate Flows

1. If the validation fails the user can try again afte fixing the issues

Exceptions

- The end time is before the start time.
- 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 displayed.

Post-conditions
The event is successfully added to the calendar and displayed in the correct time slot.

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

Table 3.11: Add Event Manually

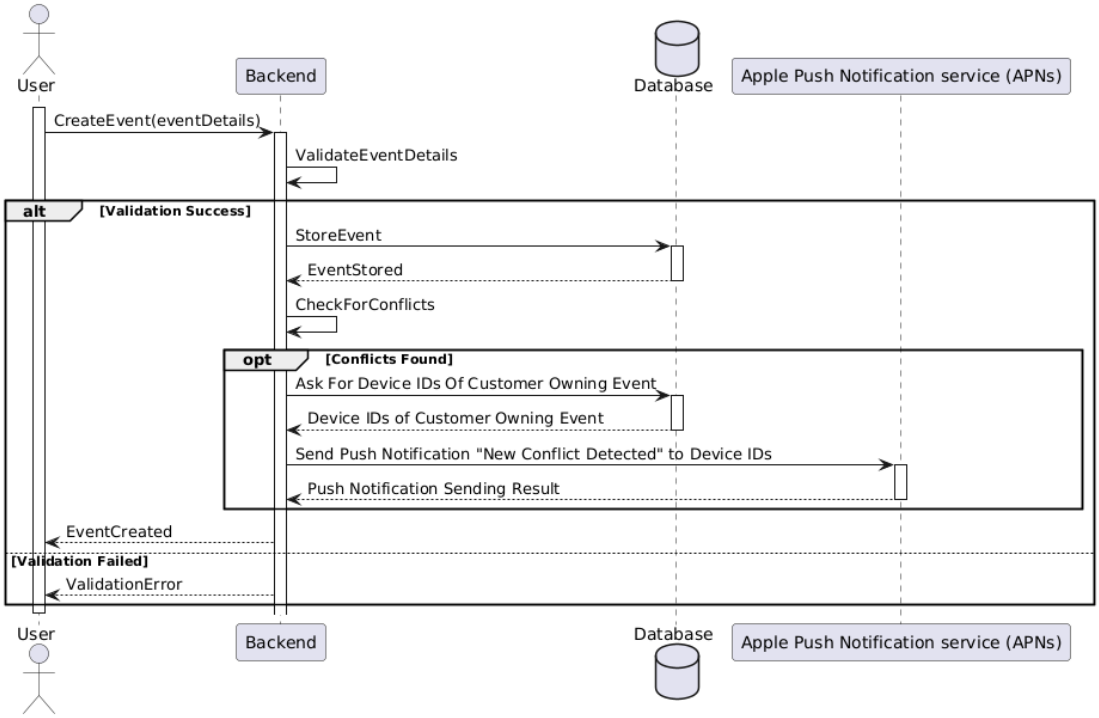

Figure 3.12: Add Event Manually Sequence Diagram

Use Case 12: View Integrated Calendar

Basic Information

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

Short Description

Allows users to view their integrated calendar

Trigger

User selects the option to view the integrated calendar.

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

- **Integrated calendar** (Source: System)

Major Outputs

- **Integrated calendar** (Destination: App)

Main Flow

1. The user clicks on the integrated calendar.

   Information: The system displays the all integrated calendars and it's events

Alternate Flows
N/A

Exceptions
• **N/A**

Conclusion
User successfully views their integrated calendar with all the events.

Post-conditions
The integrated calendar is displayed

Special Requirements
N/A

Business Rules
• **Events must be displayed according to user preferences .**

Table 3.12: View Integrated Calendar

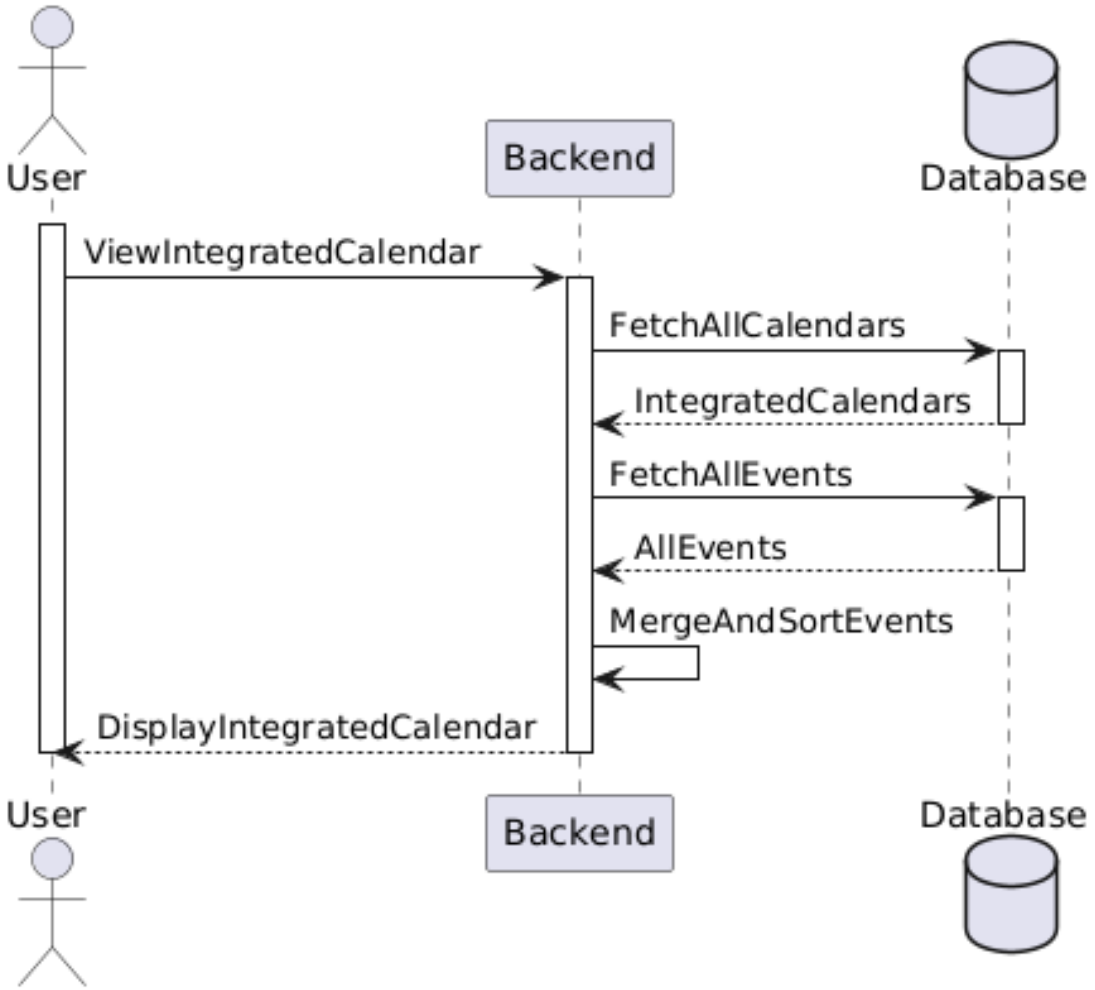

Figure 3.13: View Integrated Calendar Sequence Diagram

Use Case 13: Schedule Prayer Times

Basic Information

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

Short Description

The calendar is blocked and updated automatically according to the person's time zone prayer time.

Trigger

This usecase triggered when the user enables the prayer time feature in the application.

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 time Zone** (Source: user)
- **We get the IP address of the user and get their time zone**

Major Outputs

- **The calendar displays the blocked time for prayer time according to their time zone.**

Main Flow

1. User enables the feature by clicking the the option Schedule Prayer Time.

   Information: The system checks the time zone of the user and blocks the calendar accordingly.

Alternate Flows

- The user doesn't enable the scheduling prayer time.

Exceptions

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

Post-conditions

The system generates calendar with prayer time

Special Requirements

The system must block the calendar according to their time zone

Conclusion

User's prayer times are successfully scheduled.

Business Rules

- Prayer times must be within valid time ranges.

Table 3.13: Schedule Prayer Times

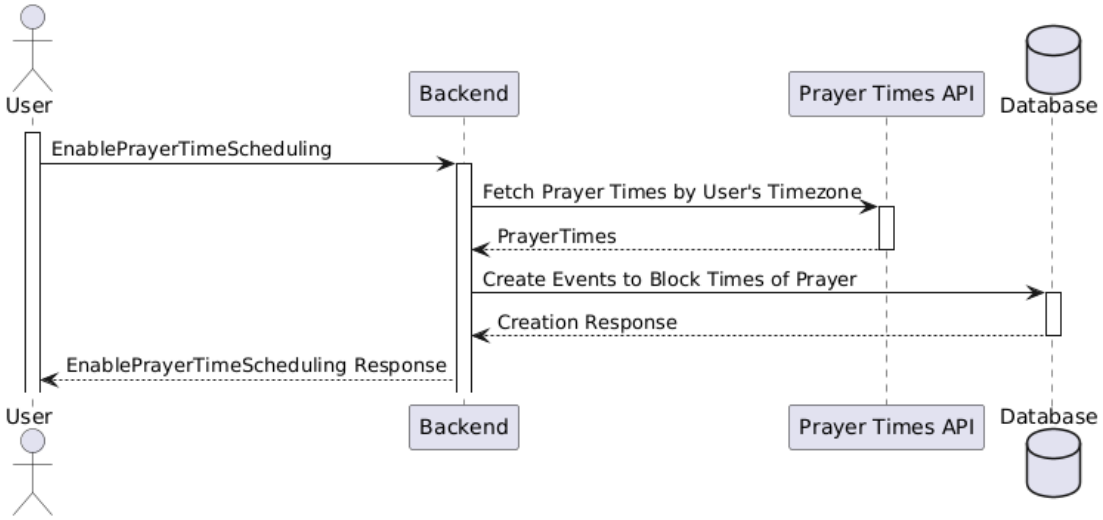

Figure 3.14: Schedule Prayer Times Sequence Diagram

Use Case 14: Receive Event Notifications

Basic Information

ID Number: 14 **Priority:** High **Type:** Regular

Short Description

Users receive notifications about upcoming events.

Trigger

The UC is triggered when the choosen time of an event's reminder has arrived

Actors

Primary: User **Secondary:** None

Preconditions

User must be logged into the system and set a reminder of the specific event.

Relationships

Extends: N/A **Includes:** N/A

Generalization/Specialization: N/A

Major Inputs

- **Time of the event reminder** (Source: User)

Major Outputs

- **Notifications sent to users.** (Destination: System)

Main Flow

1. The system checks the alarms set for every event.

   Information: The system checks every 1 minute for set alarms for every event.

2. Notifications sent to users.

   Information: The user is reminded about the event by sending the notification

Alternate Flows

- **If user denys the access to the notification the notifications are not sent.**

Exceptions

- **Netwok issue**

Conclusion

The system checks for set alarm every 1 minute and if the event is detected the system sends the notification.

Post-conditions

Notifications are sent.

Special Requirements

Notification permission.

Business Rules

The system has to check every minute for the set alarm.

Table 3.14: Receive Event Notifications

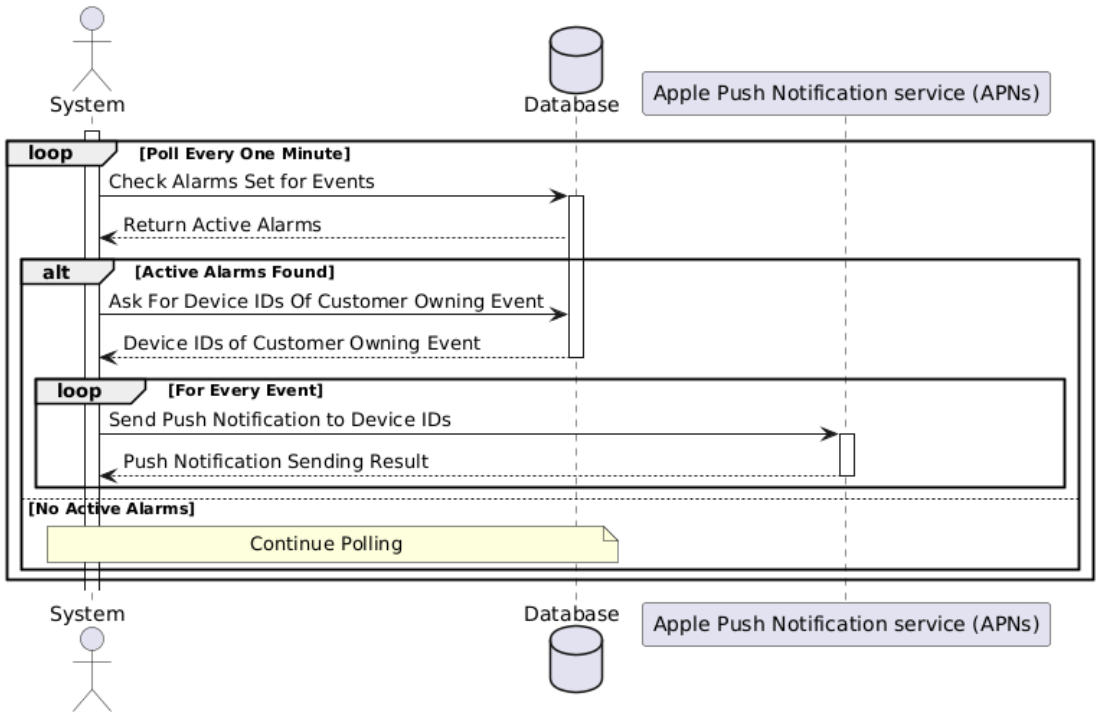

Figure 3.15: Receive Event Notifications Sequence Diagram

3.5 Activity Diagram

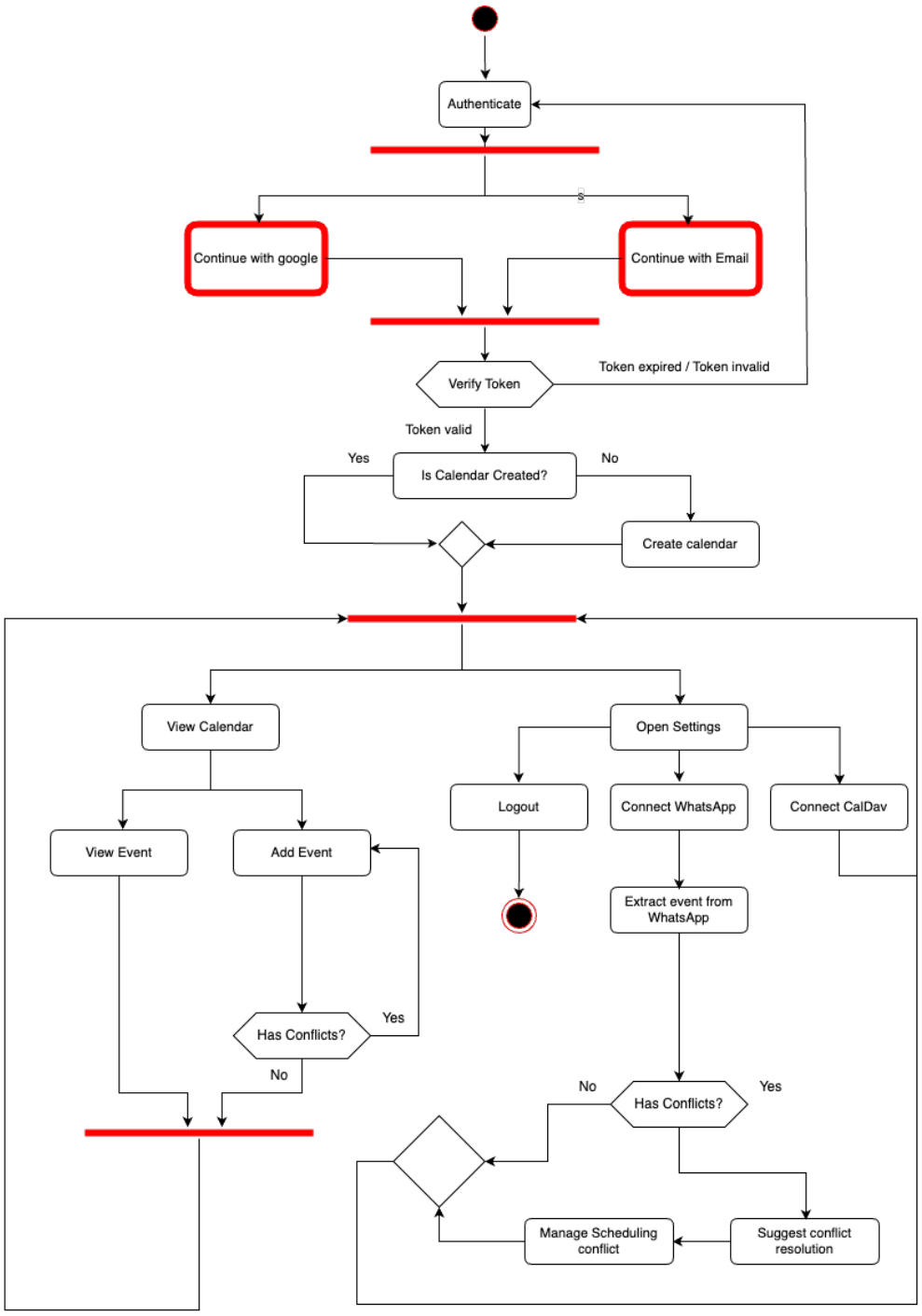

Figure 3.16: Activity Diagram of Jadwal

An activity diagram is shown in Figure 3.16 above, and it illustartes the flows the user of the app can take. It starts with the authentication whcih supports both Google and Email. Then afater the token is verified, a check is done to see if the user has a calendar or not. If the user has a calendar, we just move on, but if they don't have one, we create one and then move on. Here the user is inside the app and has two options, he can either view the calendar or open the settings page. When he views the calendar, he can view an event, or add one. Adding event might result in a conflict, so if one arises, they can try editing it and adding it again til no conflicts arise. After any of those two steps, the user can go back to view calendar or open settings page. If the choose to open settings page, they have three actions available. They can connect a CalDAV account, connect WhatsApp, or logout. Starting with the connecting a WhatsApp account, the system will extract events from the user's WhatsApp account automatically. If any conflicts arise, the system suggests a conflict resolution, and the user can take action by managing the scheduling conflict. But if no conflicts arise, the system just continues normally. Another option for the user is connecting a CalDAV account, and once that happens, the user can go back to having the option of going to calendar view or open settings page. Finally, the logout action and that terminates the session and the user is logged out of the app.

3.6 Class Diagram

The figures below show the main classes Jadwal's system includes. The figures give an overview of interfaces in the system to help in understanding the different parts of the system. Since Golang will be used for the backend mainly, the diagrams below are more suited for it.

Of course that doesn't mean they can't be used for implementing in other languages, but the way it was written took into consideration the language features. Golang is neither fully object oriented neither fully functional. It has `interface` and `struct`, and has implicit implementation. Implicit implementation means as long as a struct has the methods that an interface expects, that struct implemented the interface. Below is a key for the class diagrams coming below:

- S - `struct` (Go struct type)
- T - `type` (Go type definition)
- C - `class` (Go interface implementations)
- I - `interface` (Go interface definition)

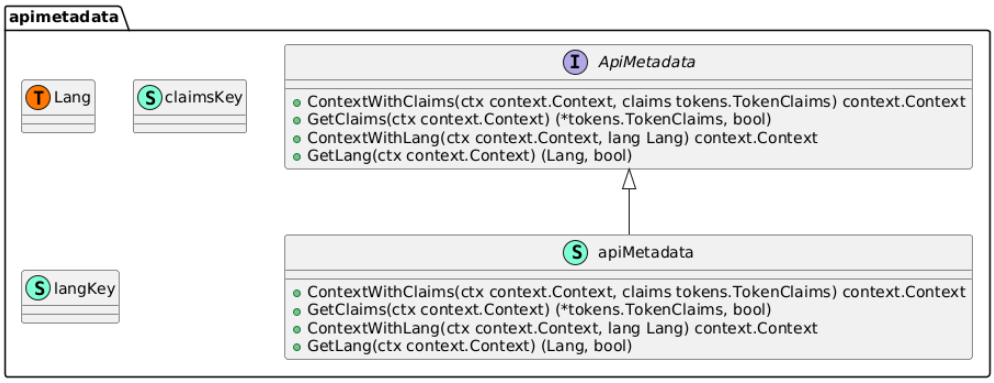

Figure 3.17: Api Metadata Class Diagram

In Figure 3.17, the diagram shows the Api Metadata interface and its implementation along with important fields. It basically extracts data from the headers and includes it in the context of the requests. It helps you put stuff into the context and extract it from them. Context in Golang allows you to attach stuff to it, but it is not type safe, so we are making a wrapper to make it safe. Below we are adding wrappers for `Claims` and `Lang`. `Claims` are extracted from the JWT (JSON Web Token). `Lang` is self-explanatory.

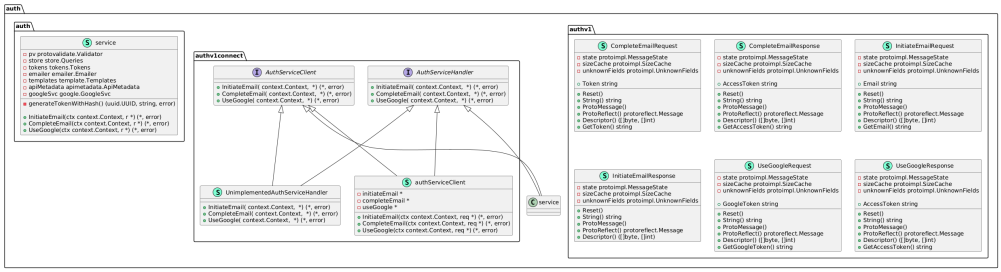

Figure 3.18: Auth Class Diagram

In Figure 3.18, the diagram shows the Auth API interface with its dependencies. It also shows the `authv1connect` which is the generated code for the API. This includes the flow for using email to authenticate and using Google.

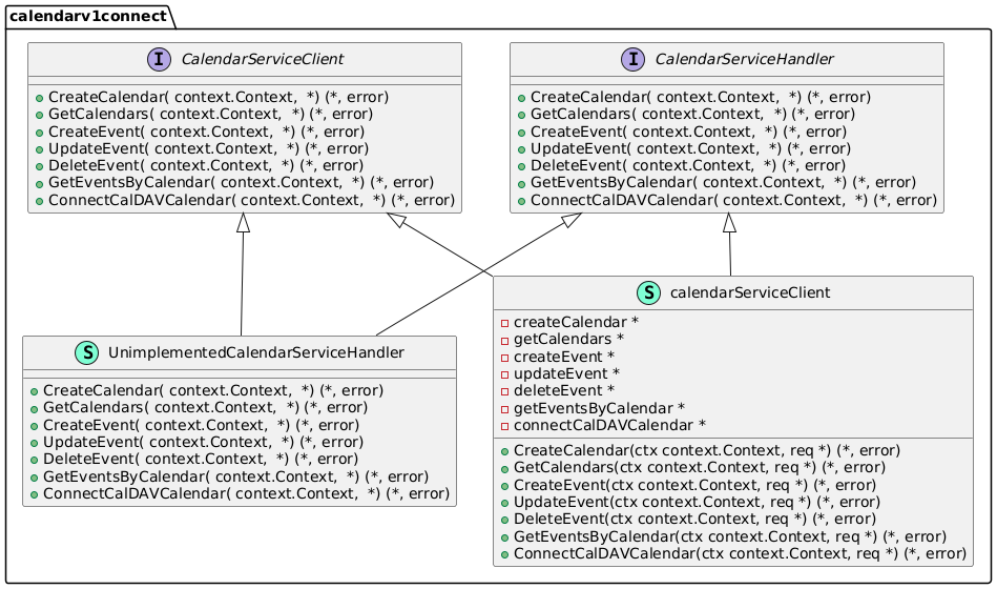

Figure 3.19: Calendar V1 Class Diagram

In Figure 3.19, the diagram shows the api layer service calendar. It allows Jadwal's clients to manage their calendar data easily. It shows the interfaces and the implementation structs with their fields and methods.

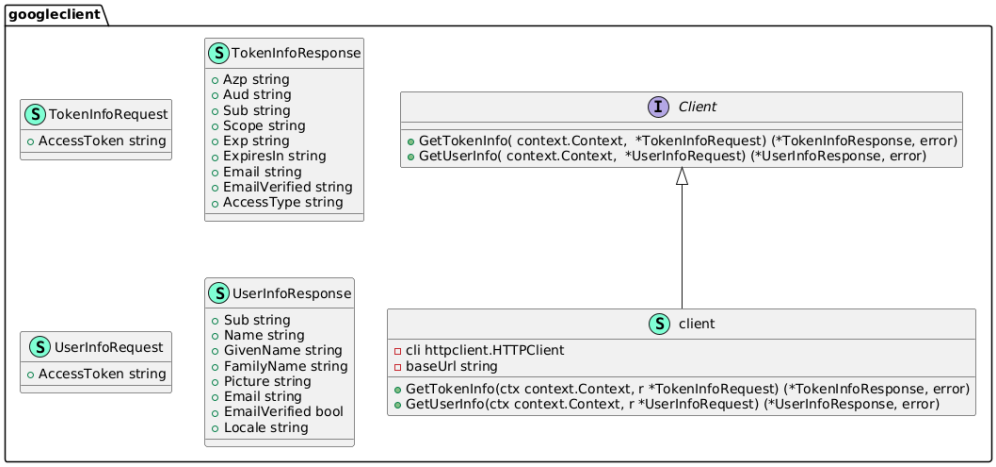

Figure 3.20: Google Client Class Diagram

In Figure 3.20, the Google client, and its methods. It allows Jadawl to verify with Google that tokens its gets are good, and it also allows Jadwal system to get the user info the system was granted access for.

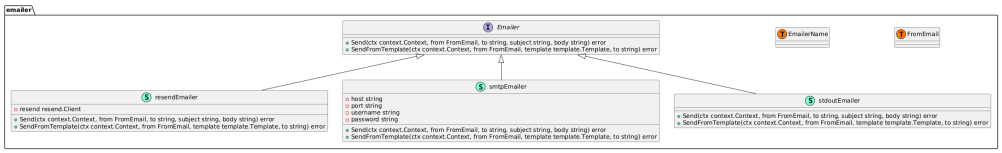

Figure 3.21: Emailer Class Diagram

In Figure 3.21, the diagram shows the `emailer` interface with its three implementations. This allows for using the textttstdoutEmailer when testing locally, and using either of the `resendEmailer` or `smtpEmailer` in production to send real emails.

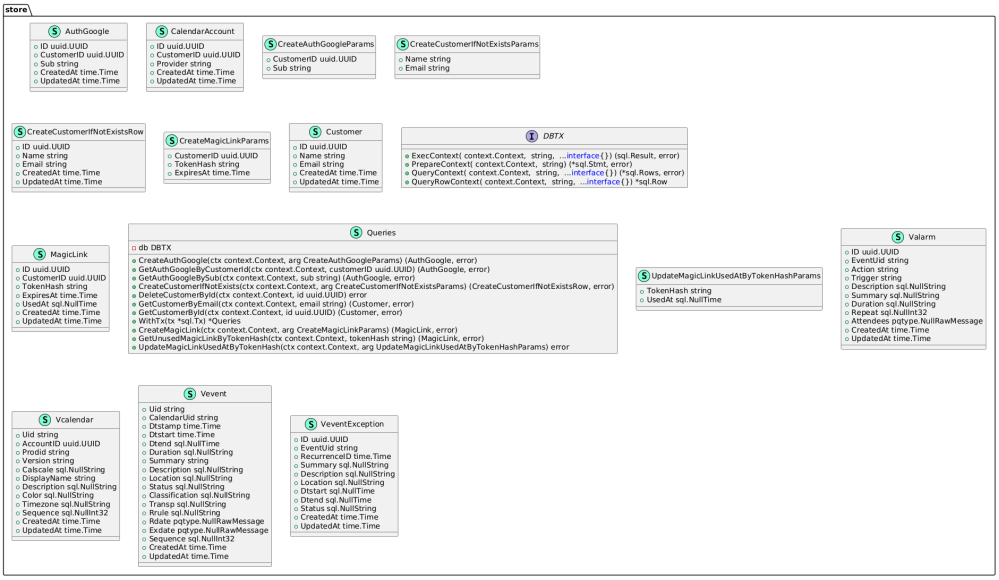

Figure 3.22: Store Class Diagram

In Figure 3.22, the diagram shows the store, or in other words the database. The store holds all the methods needed to query the database for information it holds. More methods can be added as needed, but those are the core methods.

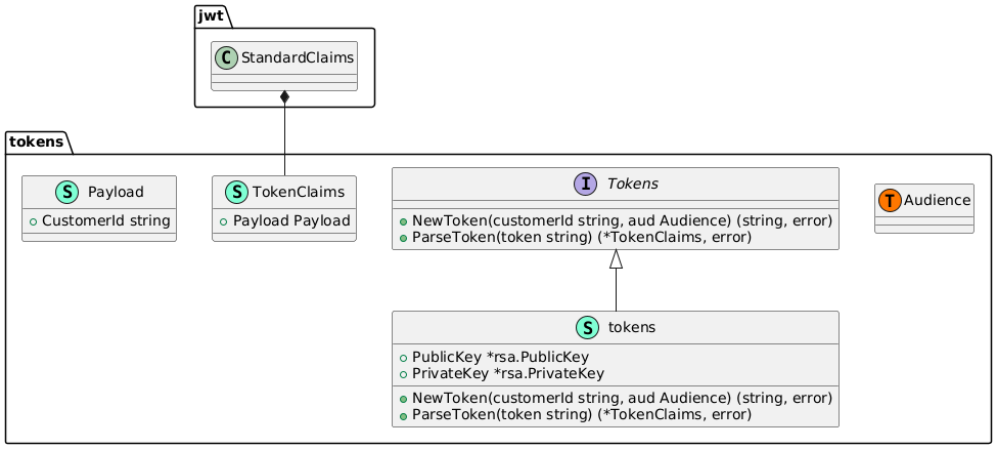

Figure 3.23: Tokens Class Diagram

In Figure 3.23, the diagram shows the tokens service implementation along with its interface. The diagram also illustartes the needed `struct` representation for any data passed to and from the service.

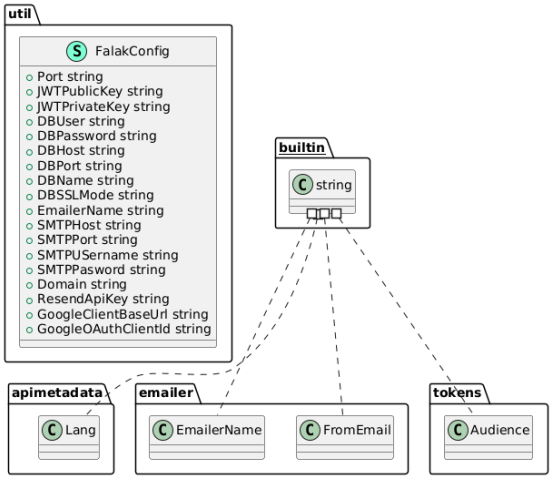

Figure 3.24: Util Class Diagram

In Figure 3.24, the diagram shows config needed to support both production and development without committing secrets to the codebase. Falak in this context is the codename for the backend. The config here is represented as a `struct` since it is hold data and doesn't need implementations.

3.7 Database Design

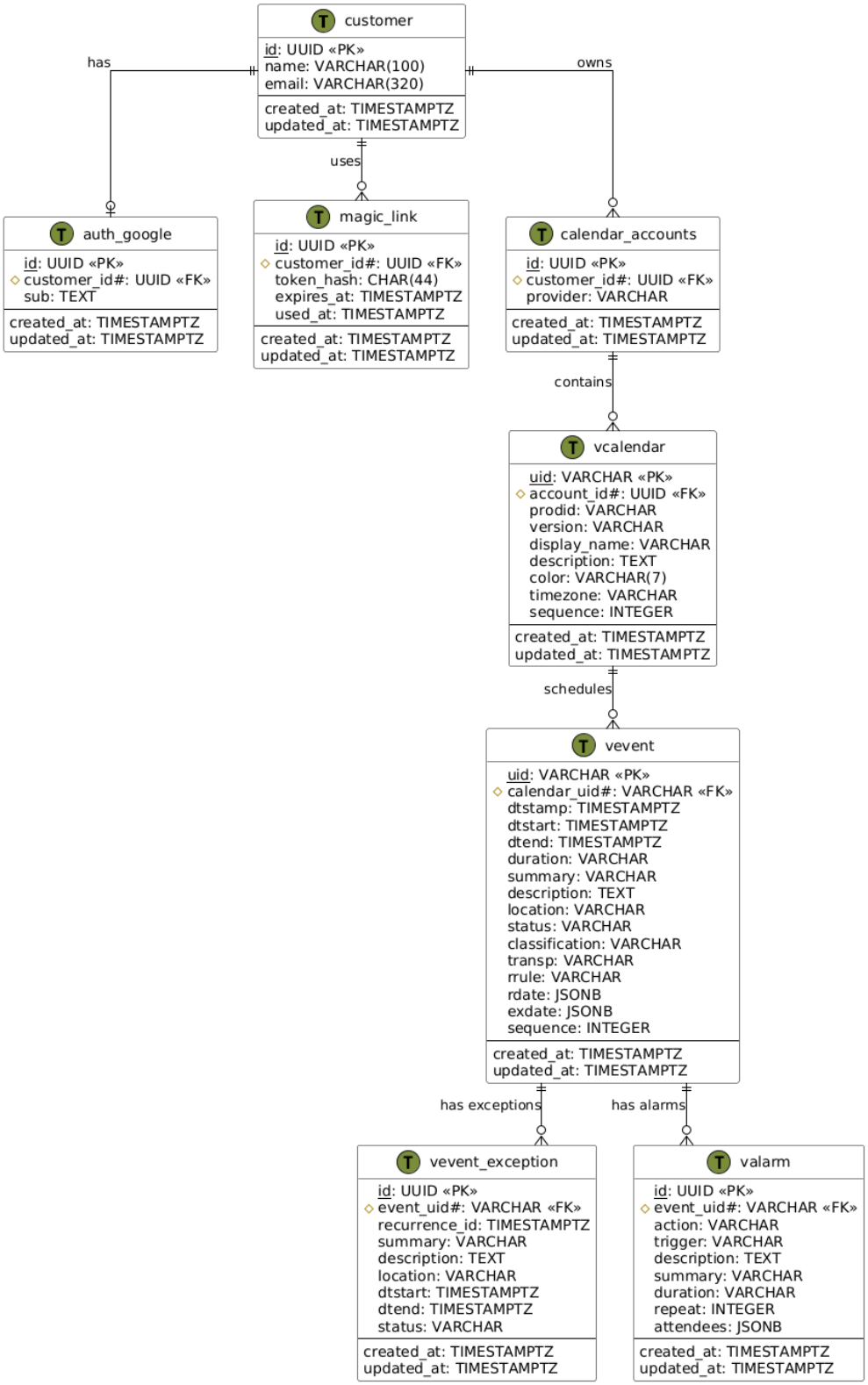

Figure 3.25: Database Design

In Figure 3.25, the tables and the relations between them in Jadwal's system is shown. The `customer` is the main table. Three tables have foreign keys that reference it. The `auth_google` and `magic_link` tables are used for auth, and they both save the info required to the authentication and hint at which authentication methods are allowed for a specific customer.

A third table related to the `customer` table is the `calendar_accounts` table. It makes it possible to link multiple calendars to one customer under a calendar account. Then the heirarchy continues, each `calendar_accounts` entry contains many `vcalendar` entries. And each `vcalendar` entry contains many `vevent` entries. And each `vevent` entry has exceptions and alarms related to it
via `vevent_exception` and `valarm` respectively.

3.8 User Interface Prototype

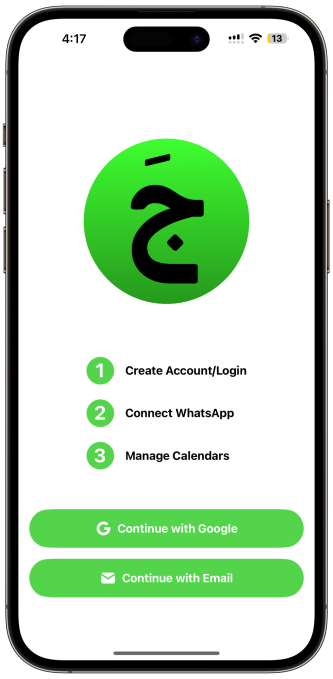

Figure 3.26: UI Screen 1: Onboarding View

In Figure 3.26, the screen shows the user the steps he needs to take in the future so he knows what he is going to do. It also has two way of authenticating, Google and Email (Magic Link).

In Figure 3.27, the screen is showing the continue with email screen which allows a user to enter their email, click ”Continue”, and receive an email if all is good.

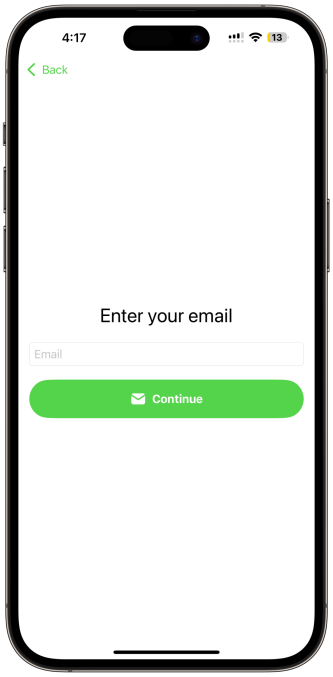

Figure 3.27: UI Screen 2: Continue with Email View

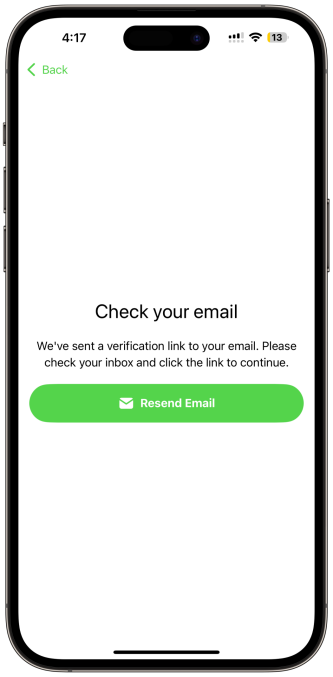

Figure 3.28: UI Screen 3: Check Your Email View

In Figure 3.28, the screen that tells the user to check their email for the magic link is shown. They can also click ”Resend Email” to get another copy if the first one wasn't received. If they received it, they can click the link inside it and it will redirect back to the app and the app will be able to read the url and its contents which has a token which the app uses to finish the authentication process. Once the authentication process is done, the user is moved to the screen shown in Figure 3.29

In Figure 3.29, the calendar view represents the user's schedule. It shows upcoming events, past events, and current events. It also has two buttons at the top, notifications and add event buttons. Users can swipe left and right to navigate months. The current day is colored in green.

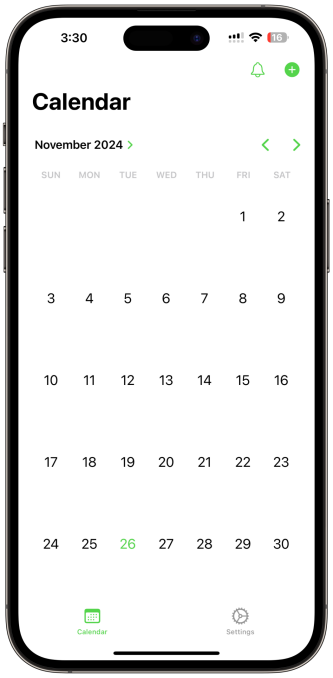

Figure 3.29: UI Screen 4: Calendar View

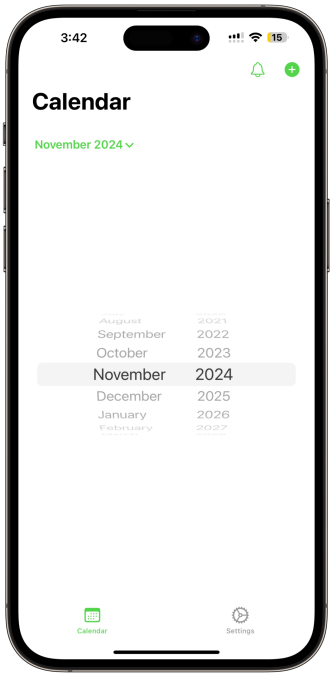

Figure 3.30: UI Screen 5: Month & Year Selector

In Figure 3.30, users can click on the text that shows the currently selected month, in this case “November 2024” and get a selector wheel that allows them to choose a month and a year to navigate to.

In Figure 3.31, the screen is showing the add event sheet in its default state. It is shown when you click the ”add event” button in Figure 3.29.

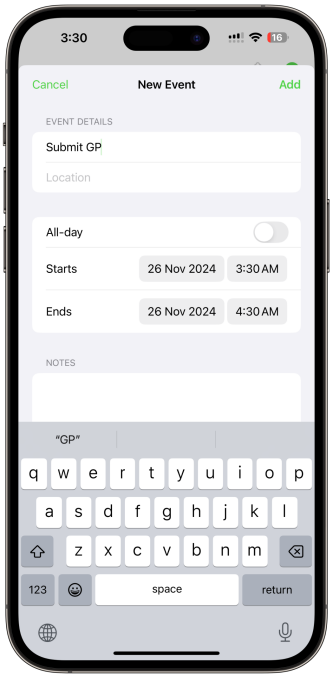

Figure 3.31: UI Screen 6: Add Event View - Default

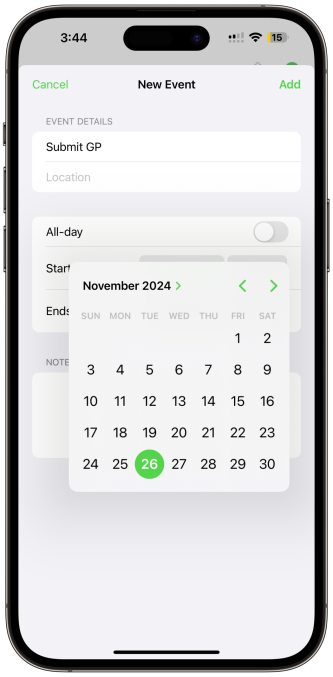

Figure 3.32: UI Screen 7: Add Event View - Date Picker

In Figure 3.32, the screen is showing the add event sheet in its date picker chosen state. You can choose a date and scroll between months and years if you want.

In Figure 3.33, the screen is showing the add event sheet in its time picker chosen state. You can choose a time and scroll between hours and minutes if you want.

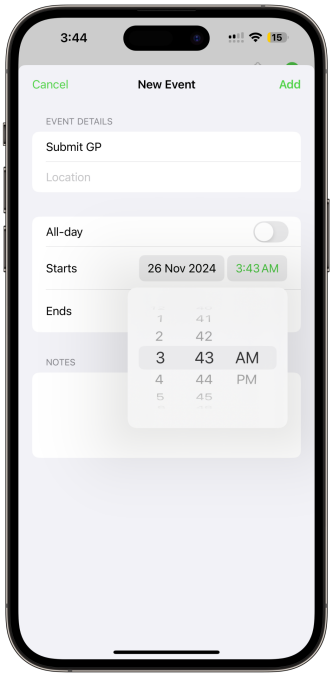

Figure 3.33: UI Screen 8: Add Event View - Time Picker

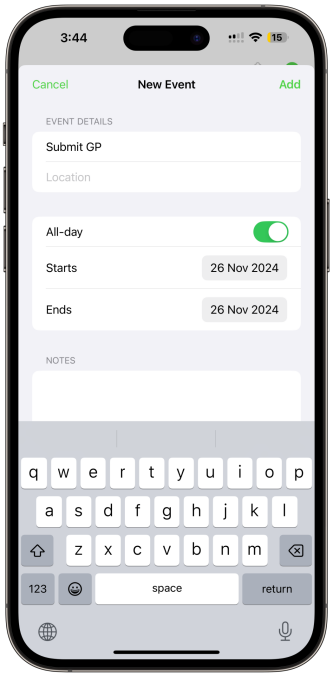

Figure 3.34: UI Screen 9: Add Event View - All Day

In Figure 3.34, the screen is showing the add event sheet in its all day state. That means you only choose the dates, no need for the times.

In Figure 3.35, the settings page is shown. This pae has the user details, specifcally email, and name. Also the ”Connect WhatsApp” button is shown along with the ”Connect CalDAV” button. Those two buttons do as they say and allow users to have data sources for the calendar connected. The last button shown is the logout button, and this button is in red to make the user alerted and not click it by mistake. Clicking it will log the user out and take them to the home screen.

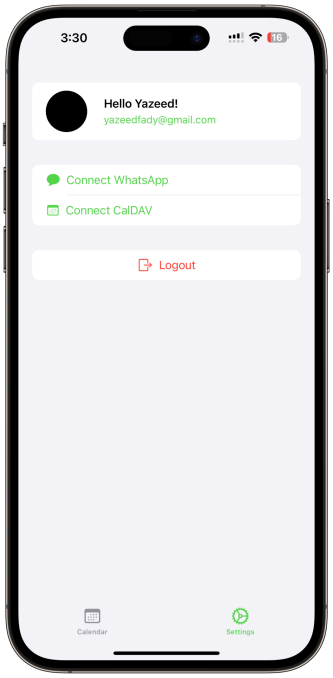

Figure 3.35: UI Screen 10: Settings View

3.9 Conclusion

Jadwal stands as a comprehensive and innovative solution for scheduling with great design. The use cases and database architecture are well described that helps in creating the application systematically. The use cases clearly define how users interact with the system, covering all the features like Continue with email, Resolve conflicts, WhatsApp integration, scheduling prayer time and many more. The database design is the backbone of Jadwal, which clearly shows how the system works and how the data is managed perfectly. In conclusion, Jadwal is well described which makes it easy to understand how everything is working and would be easy to track everything that is happening in the backend.

Bibliography

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

[Cambridge, 2024] Cambridge (2024). Calendar definition. Accessed: 2024-10-29.

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

[Golang, 2024] Golang (2024). Golang definition. Accessed: 2024-10-30.

[JWT, 2024] JWT (2024). Jwt definition. Accessed: 2024-10-30.

[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.

[WhatsApp, 2024] WhatsApp (2024). Whatsapp about page. Accessed: 2024-10-29.