جامعة اليمامة
Al Yamamah University

AL YAMAMAH UNIVERSITY

College of Engineering and Architecture

Bachelor of Science in Software and Network Engineering

Jadwal: An Intelligent, 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: 3 Dec 2024
Supervised By: Dr. Inayatullah

First Semester 2024–2025

ABSTRACT

With rapid globalization, time management has became a real challenge. This paper introduces Jadwal, which proposes an automatic schedule management system where an individual can integrate multiple calendars and extract the events that have been discussed in 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 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. Inayatullah for his invaluable guidance and precious time. His advise was instrumental in shaping our project. He provided support in challenging times. Moreover, he emphasized that all members should be involved in each phase of the project and should gain core knowledge and contribute effectively.

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

This chapter provides a comprehensive list of technical terms, abbreviations, and specialized vocabulary used throughout this document. Understanding these terms is crucial for comprehending the technical aspects of Jadwal's architecture, functionality, and implementation. The following table defines each term in clear, accessible language.

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
WebDAV	Web Distributed Authoring and Versioning (WebDAV) consists of a set of methods, headers, and content-types ancillary to HTTP/1.1 for the management of resource properties, creation and management of resource collections, URL namespace manipulation, and resource locking (collision avoidance). [WebDAV, 2007]
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]
gRPC	A high-performance, open-source universal RPC framework that enables efficient communication between services. [gRPC, 2024]
Protocol Buffers	A language-neutral, platform-neutral extensible mechanism for serializing structured data. [Protobuf, 2024]
RabbitMQ	An open-source message broker that enables asynchronous communication between services. [RabbitMQ, 2024]
PostgreSQL	A powerful, open-source object-relational database system

	used for storing application data. [PostgreSQL, 2024]
SwiftUI	Apple's declarative framework for building user interfaces across all Apple platforms. [SwiftUI, 2024]
WhatsApp Web.js	A WhatsApp client library that enables programmatic interaction with WhatsApp Web. [WhatsAppWebJS, 2024]
Node.js	A JavaScript runtime built on Chrome's V8 JavaScript engine. [NodeJS, 2024]
HTTP/2	The second major version of the HTTP protocol, offering improved performance and features. [HTTP2, 2024]
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 (people, objects, concepts, or data) relate to each other within a system or database.
Sequence Diagram	An interaction diagram that details how operations are carried out.
APIs	Application programming interface.
APNs	Apple Push Notification service
Magic Link	A link sent via email that allows for password-less sign in.

1 INTRODUCTION

Time management has become increasingly complex in our interconnected world, where individuals juggle multiple calendars, commitments, and communication channels. This chapter introduces Jadwal, an intelligent iOS-based calendar management system designed to address these modern challenges. We will explore the background, objectives, scope, and significance of this project, highlighting how it aims to revolutionize personal time management through intelligent automation and integration with communication platforms.

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 activities.

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 communication 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 cumbersomeness of adding events to a calendar manually. And when people are busy, they just forget that they didn't add the event to their calendar. Our Jadwal app aims to solve this issue for users in an intelligent way that makes it seamless to manage your time confidently.

1.3 Objectives of the Project

The development of Jadwal is driven by clear, measurable objectives that address the current challenges in calendar management. These objectives focus on creating an intelligent system that seamlessly integrates multiple calendars, automates event extraction, and prioritizes user efficiency while maintaining prayer schedules. Each objective has been carefully defined to ensure the final product delivers meaningful value to users.

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 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 using state-of-the-art LLMs (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

In today's fast-paced environment, effective time management is not just a convenience—it's a necessity. Jadwal addresses critical gaps in existing calendar solutions by introducing innovative features that align with modern users' needs. The project's significance extends beyond simple calendar management, offering transformative benefits for personal productivity, religious observance, and professional organization. By

tackling the challenges of fragmented schedules and missed commitments, Jadwal represents a significant advancement in how people manage their time in the digital age.

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

While Jadwal introduces innovative solutions to calendar management, it is essential to acknowledge and understand its current constraints and limitations. These limitations stem from various factors including technological boundaries, privacy concerns, third-party dependencies, and resource constraints. By identifying these limitations early in the development process, we can better manage user expectations and plan future improvements to address these challenges.

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 user's messages, so it would be difficult to prove that 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.
- 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

To ensure systematic development and timely delivery of Jadwal, we have established a detailed project timeline with clear milestones and deliverables. The Gantt chart, shown in **Figure 1.1**, illustrates our project's phases, tasks, and their respective durations.

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.

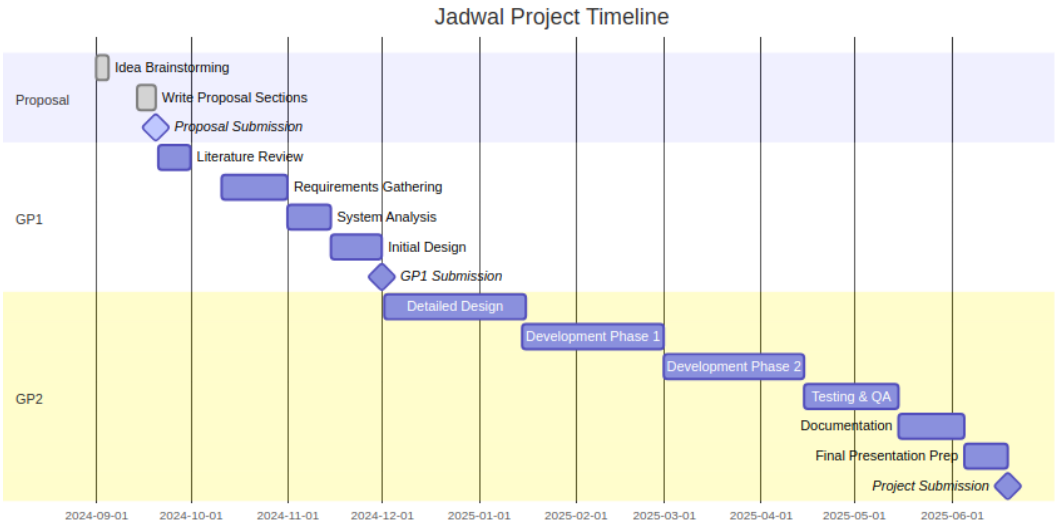

Figure 1.1: Project Gantt Chart

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 or have conflicts.

2 LITERATURE REVIEW

The landscape of calendar management and scheduling applications has evolved significantly with technological advancement and changing user needs. This chapter presents a comprehensive analysis of existing solutions, market research, and technological foundations that inform Jadwal's development. Through examining current tools, academic research, and user behavior patterns, we identify gaps in existing solutions and validate Jadwal's innovative approach to calendar management.

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 stands out 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 re-

search 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 support 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

To thoroughly understand Jadwal's position in the market and its unique value proposition, a detailed analysis of existing calendar management solutions is essential. This section examines four major competitors—Clockwise, Motion, Reclaim AI, and Calendi—evaluating their features, strengths, and limitations to highlight how Jadwal addresses gaps in current offerings.

2.2.1 Clockwise

Clockwise's website tries to touch the user's misery by saying "Your calendar doesn't have to suck" as shown in Figure 2.2.

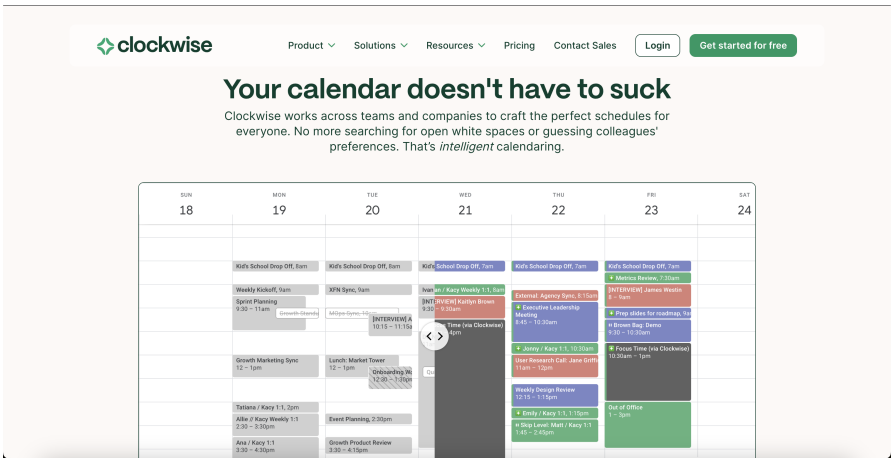

Figure 2.2: Clockwise Landing Page

Based on Figure 2.3, Clockwise has the following features:

- Connect your calendar and set your preferences
- Decide which meetings are flexible
- Let Clockwise optimize your day

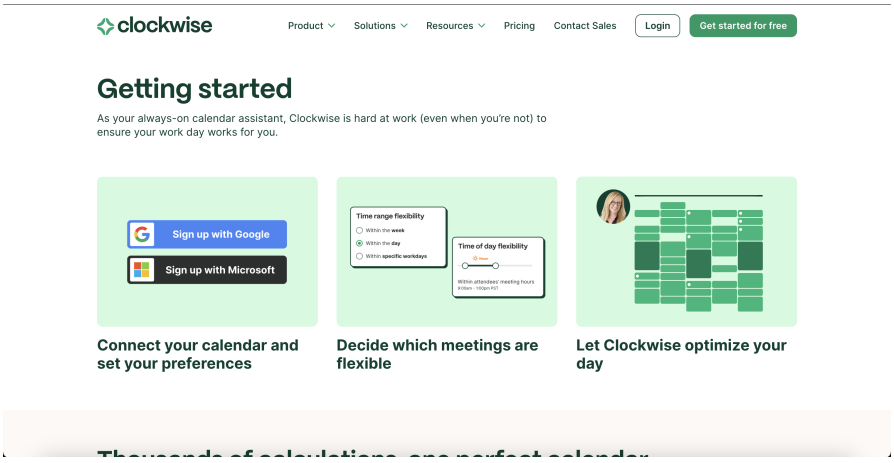

Figure 2.3: Clockwise Features Page

2.2.2 Motion

Motion's website boasts about AI features in their landing page, as shown in Figure 2.4.

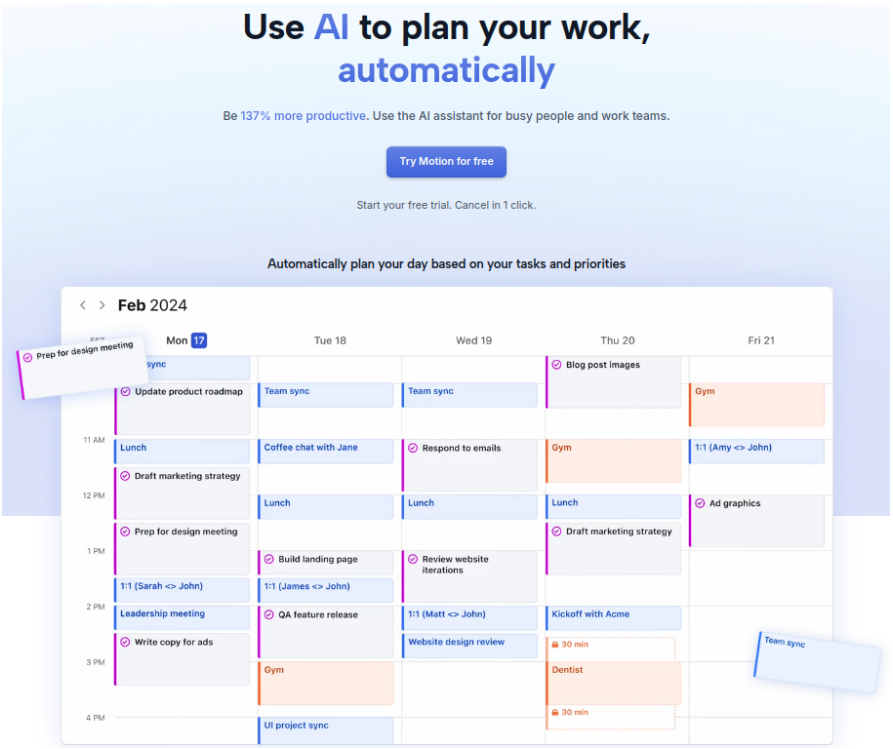

Figure 2.4: Motion's Landing Page

Its features include a dynamic calendar that automatically prioritizes tasks, suggests optimized schedules, and helps users adapt to changes seamlessly.

The tool also provides team collaboration features, allowing shared schedules and streamlined task management to enhance team productivity. These features ensure that users can achieve maximum efficiency with minimal effort.

Motion allows you to combine your calendars in one view also, shown in Figure 2.5.

Motion also has a time blocking feature which they boast about in the website. It is shown in Figure 2.6.

Jadwal platform, such as linking the calendar to WhatsApp for automatic event addition and integrating prayer times, which adds a unique level of customization and day organization.

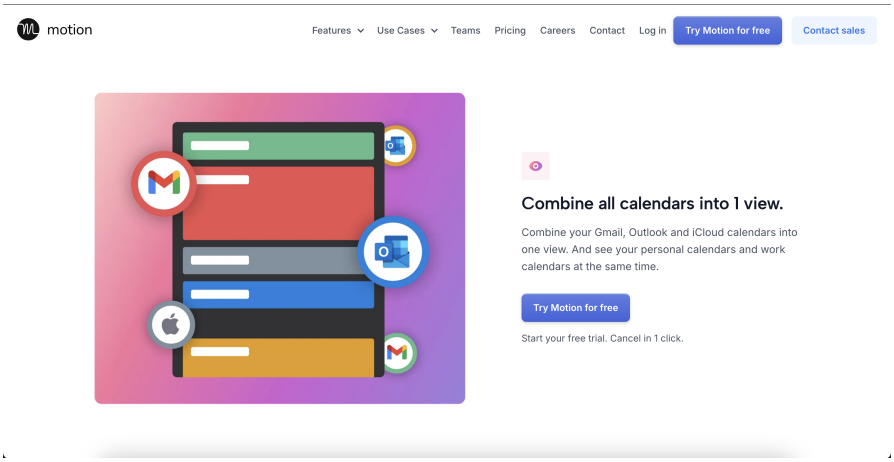

Figure 2.5: Motion's Combined Calendar Feature

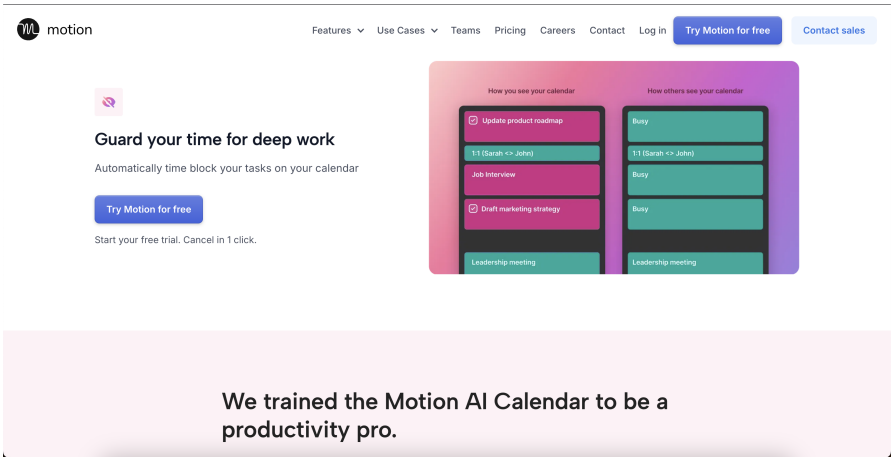

Figure 2.6: Motion's Time Blocking Feature

2.2.3 Reclaim AI

Reclaim.ai offers the following features: It is an AI-powered platform designed to manage tasks, habits, meetings, and personal events seamlessly. Figure 2.7 shows the the landing page of Reclaim and their focus on AI as you can see.

Reclaim.ai provides features such as smart time blocking, which automatically schedules tasks based on their priority and deadlines, integrates with tools like Google Calendar, Slack, and project management platforms (e.g., Asana, Jira, and Todoist), and supports habit tracking with auto-rescheduling to adapt to life changes. Figure 2.8 shows the integrations that this platform supports.

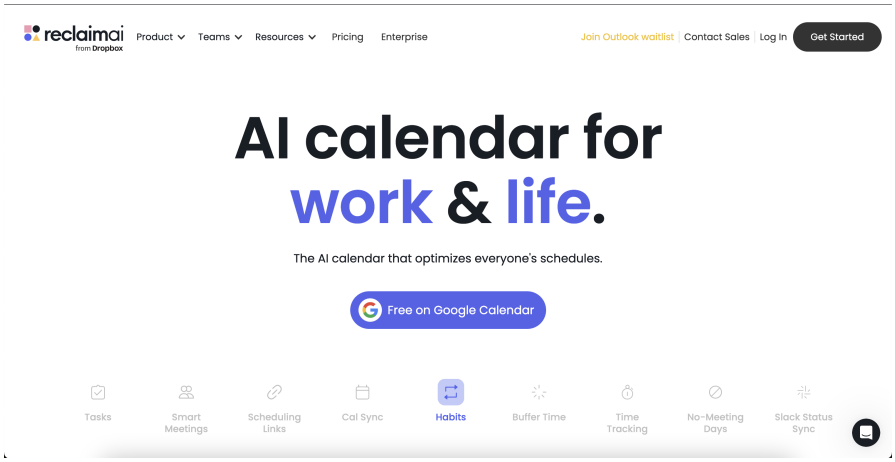

Figure 2.7: Reclaim Landing

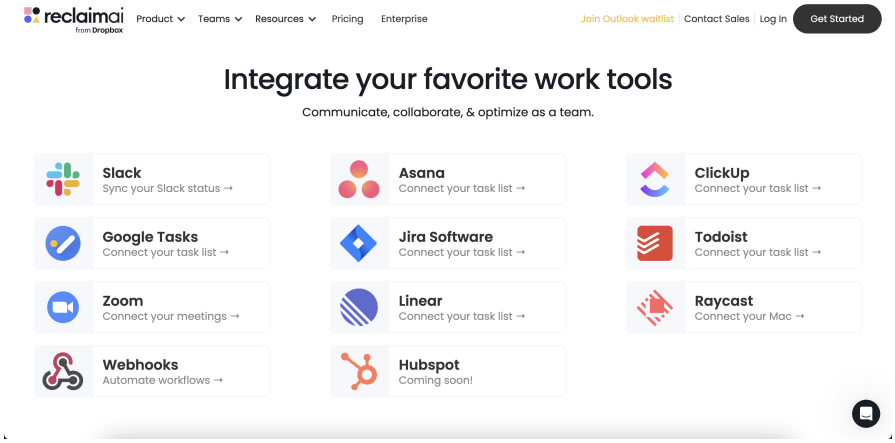

Figure 2.8: Reclaim Integrations

While these features enhance productivity and optimize time management, Reclaim.ai does not include features like WhatsApp integration for automatically adding events or prayer time scheduling, both of which are unique to the Jadwal platform. These features provide users with cultural and practical benefits that further enhance their daily organization.

2.2.4 Calendi

Calendi.ai claims to offer features focused on speeding up scheduling with AI-powered assistance.

It claims to allow users to add tasks via text or voice input and automates meetings

scheduling, including sending links and calendar invites. Those claims are shown in Figure 2.9.

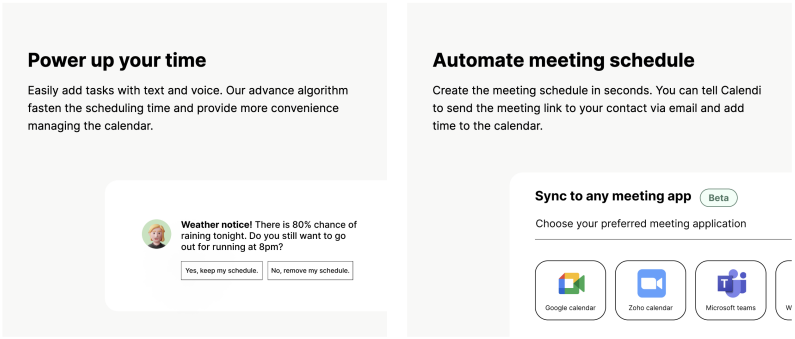

Figure 2.9: Calendi Features of Voice Input and Automatic Meeting

Calendi claims to integrate with platforms like Google Calendar, Apple Calendar, and Outlook, making it compatible across multiple systems. This is shown in Figure 2.10, as they say "We replace them all, help you save 'clock emoji' 'money emoji' and schedule better."

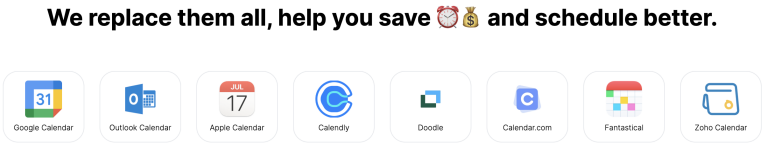

Figure 2.10: Calendi Claims to Replace Them All

It claims to provide intelligent assistance by analyzing habits and tasks, suggesting improvements, and incorporating external factors like weather or transit schedules. Those are shown in Figure 2.11.

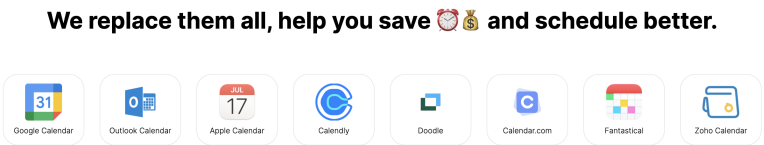

Figure 2.11: Calendi Claims to Replace Them All

However, it does not offer unique features like WhatsApp integration or prayer time

scheduling, which are distinct to Jadwal.

Also the website as of writing this part, on 26 Nov 2024, is broken and not working. This can be illustrated via the broken timer shown in Figure 2.12 and many other things that are broken.

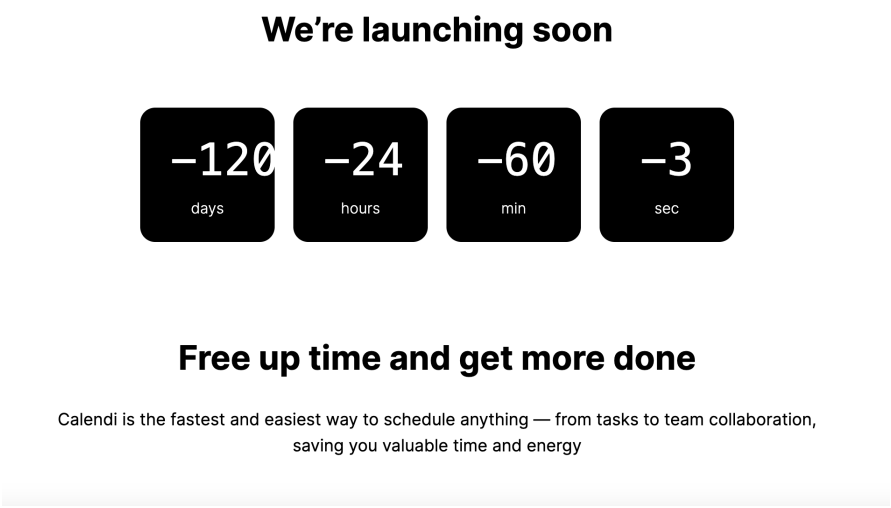

Figure 2.12: Calendi Broken Timer

2.3 Survey Results

We have conducted a survey using Google Forms, and have sent it out to our peers in the university, and also shared it on our social media, especially LinkedIn, to get more diverse responses. That is since our main audiences are students, employees, and who do both at the same time. The results were fascinating. Especially since 75 responses were recorded with their email to authenticate the responses.

As shown in Figure 2.13, the results showed that 8% only were both employed and students, while 53% were students only and 39% were employed only. We believe this is a good mix between the two audience we are aiming to get responses from.

Based on Figure 2.14, it is important to note that 72% of our respondents were of the age range 20-30 years.

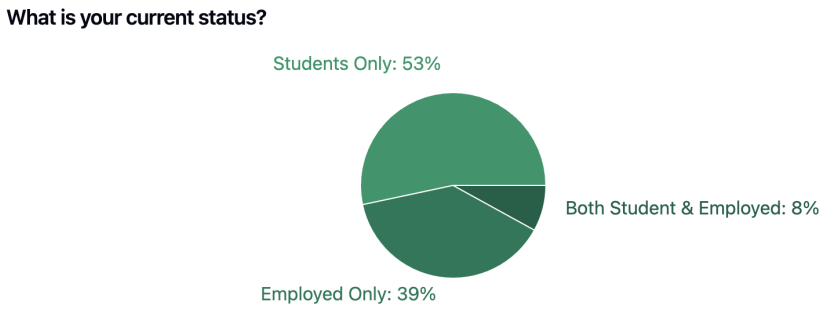

Figure 2.13: Survey Respondents Status

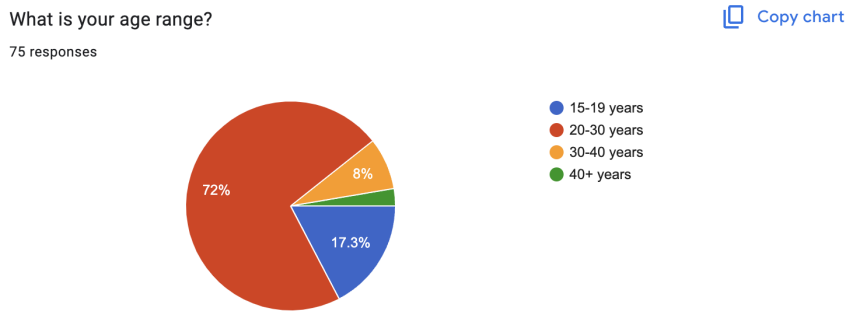

Figure 2.14: Age Range

As shown in Figure 2.15, 66.7% of people reported that they a calendar application to manage their daily schedule. This shows that people might be interested in a better way of managing their schedules that could elevate their time management skills.

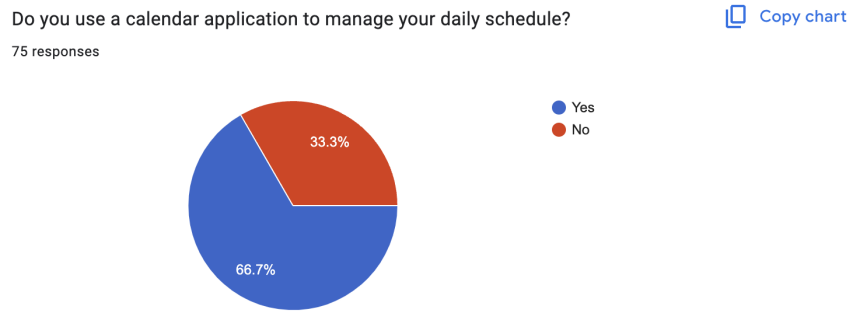

Figure 2.15: Percentage of People Who Use Calendars to Ones Who Don't

In Figure 2.16, the results showed us the need for our app to exist, since 70.7% of the population use WhatsApp to discuss upcoming events! Another big percentage, 53.3% used Email also. Our app won't be focusing on this, nevertheless, this is also an opportunity in the future to help those people who use Emails and might not find the

flow easy to schedule the events in the calendar.

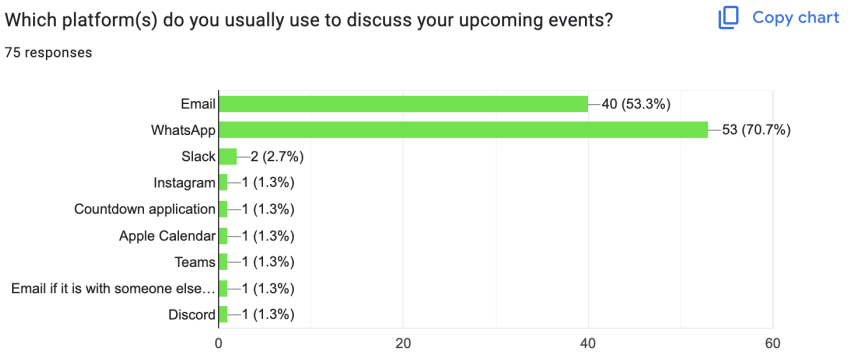

Figure 2.16: Platform Used to Discuss

Although 21.3% might not seem like a lot, but as shown in Figure 2.17, 58.7% of the population reported they would find it extremely useful to have a WhatsApp integration that automatically adds events from your conversations to the calendar?

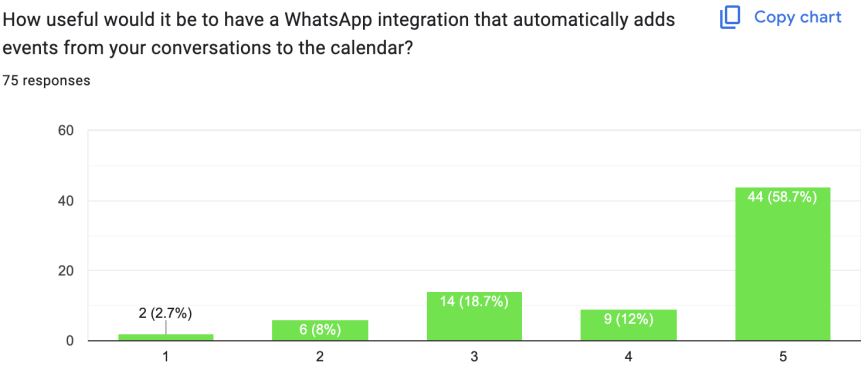

Figure 2.17: Usefulness of Having a WhatsApp Integration

As shown in Figure 2.18, on average, 21.3% forget to add events to their calendar. That is great news, since our differentiating factor is the direct WhatsApp integration that should be able to solve this.

As shown in Figure 2.19, 41.3% use 2 calendars, that shows the need for a integrated view of calendars.

As shown in Figure 2.20, 53.3% of people have 1-3 calendar events per week. That might seem a little, but 33.3% have 4-6 calendar events per week.

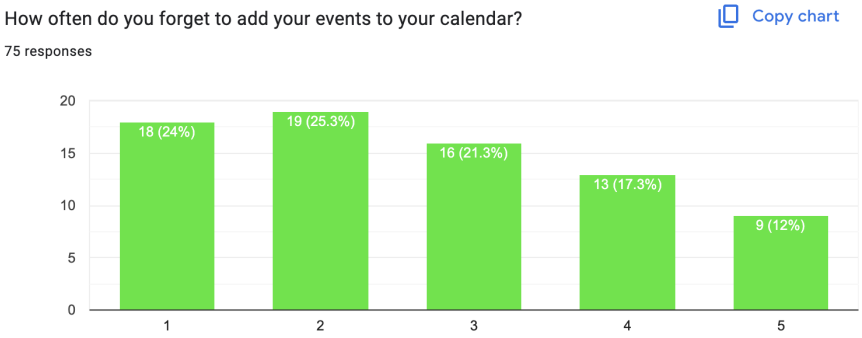

Figure 2.18: Chart of Forgetting to Add Likeliness

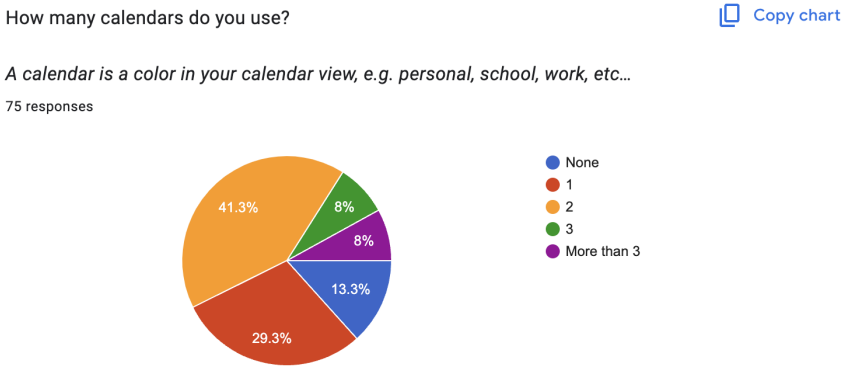

Figure 2.19: Number of Calendars

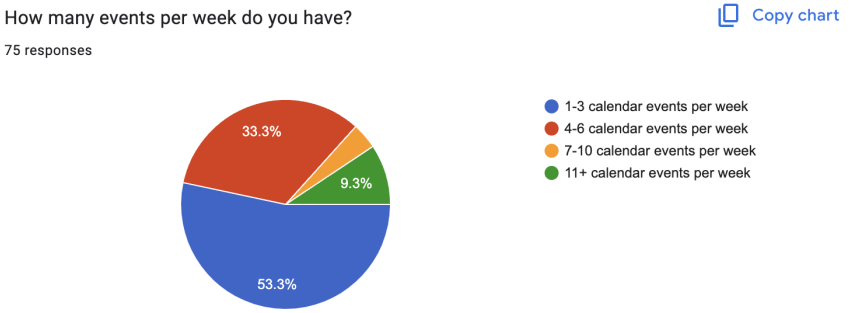

Figure 2.20: Events per Week

2.4 Conclusion

Through this literature review, we have identified several key gaps in existing calendar management solutions. While platforms like Clockwise, Motion, Reclaim AI, and Calendi have made significant strides in intelligent scheduling and calendar integration, none provide a comprehensive solution that addresses the full spectrum of modern

scheduling challenges. The review reveals three primary opportunities for innovation:

First, while existing solutions focus on traditional calendar management, there remains an untapped potential in integrating informal communication channels. Jadwal's WhatsApp integration addresses this gap, offering automated event extraction from everyday conversations—a feature notably absent in current market offerings.

Second, our analysis shows that while many applications offer conflict resolution features, none combine this with prayer time prioritization, creating a unique opportunity to serve users who prioritize religious obligations alongside professional commitments.

Finally, the review demonstrates that current solutions often operate in isolation, whereas Jadwal's approach of unifying multiple calendars while maintaining intelligent event extraction and conflict resolution presents a more holistic solution to modern scheduling challenges.

These findings validate Jadwal's approach and highlight its potential to address significant unmet needs in the calendar management space, particularly in combining automation, religious considerations, and comprehensive calendar integration.

3 SYSTEM ANALYSIS AND DESIGN

The development of a complex calendar management system like Jadwal requires careful analysis of requirements and thoughtful system design to ensure robust functionality and seamless user experience. This chapter presents a detailed examination of Jadwal's architecture, from its core requirements to the intricate relationships between system components. Through use case diagrams, activity diagrams, class diagrams, and database design (both ER and Relational), we provide a comprehensive blueprint of how Jadwal transforms its innovative concepts into a practical, functioning system.

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.

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 following requirements outline the core features and capabilities that Jadwal must provide to fulfill its purpose as an intelligent calendar management system:

- The user shall be able to access their account using either Google OAuth or magic link via Email. For new users, a new account 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.

Each functional requirement listed above will be explained in details through a use case description in the coming sections.

3.3 Non-Functional Requirements

While functional requirements define what the system does, non-functional requirements specify how the system performs its functions. These requirements focus on the quality attributes, performance standards, and technical constraints that ensure Jadwal delivers a reliable, secure, and user-friendly experience.

- **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 Security Architecture

In today's digital world, security is a key concern for any application that handles user data. Jadwal places a strong emphasis on ensuring the security and trustworthiness of the platform for its users, implementing multiple layers of security measures to protect user data.

3.4.1 Authentication: Magic Link and JWT Tokens

To provide secure authentication, Jadwal implements Magic Link authentication. Instead of relying on traditional username and password combinations, which is vulnerable to various attacks, such as credentials leakage through database dumps. To mitigate

this, Jadwal sends the users trying to authenticate a secure magic link to their verified email address. A magic link is a special URL that contains a secure token. For example:

```
https://jadwal.app/magic-link?token=some-uuid
```

In this URL, `some-uuid` is the secure token, and the complete URL is what we call the magic link. When sent to the user's email, clicking this link proves they have access to the email account they're trying to use.

When a user logs in using the Magic Link sent to their email, Jadwal secures the user's account by verifying the magic token provided. This ensures that the account can only be accessed by the legitimate user who has access to the registered email account. To enhance security, the magic token has a limited lifetime of 15 minutes, significantly reducing the window of opportunity for potential attacks.

Upon successful magic link verification, Jadwal issues a JWT (JSON Web Token) signed with the platform's private key. This digital signature serves as a cryptographic guarantee of the token's authenticity, allowing the system to verify that tokens haven't been tampered with and were legitimately issued by Jadwal.

3.4.2 Token Storage and Security

To ensure the security of authentication tokens, Jadwal implements secure storage practices for storing the magic token in the database. Magic tokens are never stored in their original form; instead, they are protected using the SHA-256 hashing algorithm before being saved in the database. When verifying user tokens, the system hashes the user-provided token and compares it with the stored hash, ensuring that even in the unlikely event of a database breach, the original tokens remain secure.

3.4.3 Secure Logout Implementation

Jadwal implements a secure logout mechanism by removing the authentication token from the user's device upon logout. This practice ensures that once a user logs out, their session token cannot be reused for unauthorized access, maintaining the integrity of user sessions.

3.4.4 Transparency Through Open Source

As part of our commitment to security and privacy, Jadwal will be released as open-source software. This transparency allows security experts and users to verify our security implementations and privacy practices. Users can inspect exactly how their data is handled and verify that our privacy commitments are upheld through code review.

3.5 System Architecture

Jadwal implements a modern, distributed architecture leveraging gRPC (Google Remote Procedure Call) for efficient communication between its components. The system is divided into several key services, each responsible for specific functionality while maintaining high performance and reliability.

The architecture, shown in **Figure 3.1**, consists of the following main components:

1. **Frontend (Mishkat)**
    - iOS SwiftUI application implementing the client-side gRPC communication
    - Handles user interface and local state management
2. **Backend (Falak)**
    - Core gRPC server implementing the primary business logic

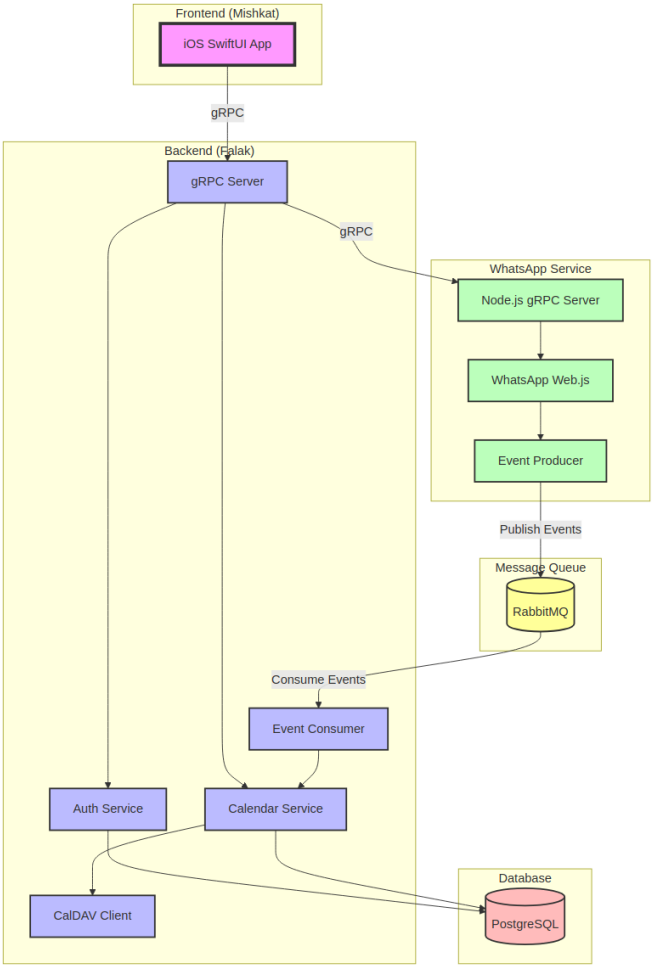

Figure 3.1: Jadwal System Architecture

- Auth Service managing user authentication and session management
- Calendar Service handling calendar operations and integration
- CalDAV Client enabling connection to external calendar services
- Event Consumer processing WhatsApp events from the message queue

3. **WhatsApp Service**
   - Node.js gRPC server managing WhatsApp communication
   - WhatsApp Web.js client for message monitoring
   - Event Producer publishing detected events to the message queue
4. **Message Queue**
   - RabbitMQ handling asynchronous event processing
   - Ensures reliable delivery of WhatsApp events to the backend
5. **Database**

- PostgreSQL storing user data, calendar information, and events
- Maintains data consistency across all services

This architecture enables several key benefits:

- **Performance**: gRPC's use of Protocol Buffers and HTTP/2 ensures efficient communication between services
- **Scalability**: Separate services can be scaled independently based on load
- **Reliability**: Message queue ensures no events are lost during processing
- **Maintainability**: Clear separation of concerns makes the system easier to maintain and update

3.6 System Use Cases

The functionality of Jadwal can be best understood through its various use cases, which demonstrate how users interact with the system. Each use case details specific interactions and flows that make up the core functionality of Jadwal. The diagram in Figure 3.2 provides an overview of all use cases and their relationships.

Figure 3.2 shows the complete use case diagram for Jadwal's system, illustrating the relationships between these fourteen distinct use cases and their actors.

3.6.1 Authentication and User Management

User authentication is the first interaction point with Jadwal. We support both email-based authentication through magic links and Google OAuth to provide secure and convenient access options. The following use cases detail the login flows and account management features.

Figure 3.2: 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

- The user cancels the authentication request.

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

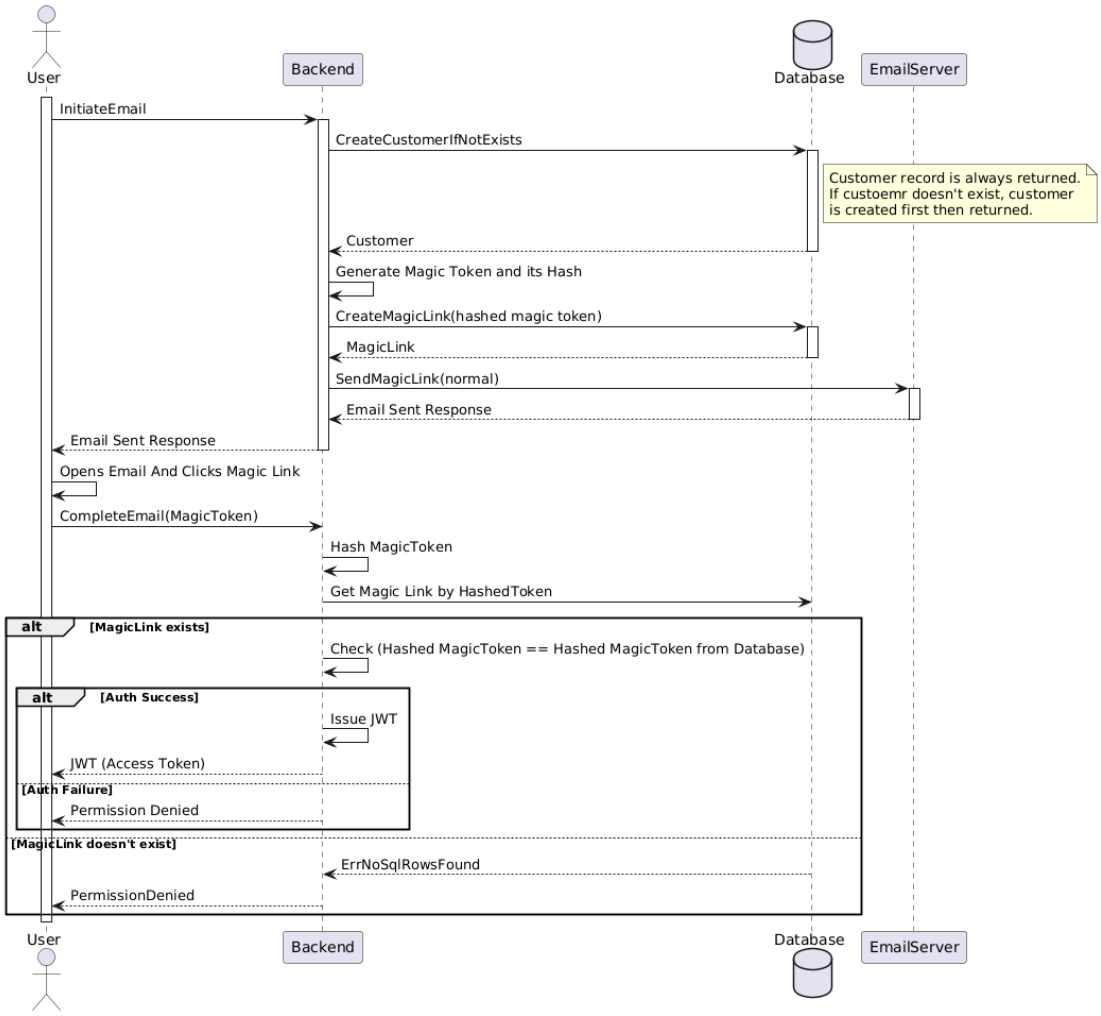

Figure 3.3: Continue with Email Sequence Diagram

The "Continue with Email Sequence Diagram", shown in **Figure 3.3**, illustrates the process of email-based magic link authentication, involving interactions between the User, Backend, Database, and EmailServer. The process begins when the user initiates email authentication process. The backend executes the CreateCustomerIfNotExists function to retrieve an existing customer record or create a new one if none exists. Once the customer is identified, the backend generates a magic token and its hashed version using the GenerateMagicToken function. The hashed token is stored in the database, and a magic link containing the token is created. The backend then sends the magic link to the user's email using the SendEmail function via the email server. The user clicks the magic link, triggering the CompleteFlow(MagicToken) request to the backend, where the provided token is validated against the stored hashed token in the

database. If the tokens match, authentication succeeds, and the backend issues a JSON Web Token (JWT) to the user for future access. In cases where the token is invalid, expired, or the customer record is missing, the system responds with appropriate errors, such as PermissionDenied or EntryNotFound. This diagram demonstrates a secure flow for handling authentication via email magic links.

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

The "Continue with Google Sequence Diagram", shown in **Figure 3.4**, illustrates how users can authenticate with Jadwal using their Google account. The user first obtains an access token from GoogleAuth through our application and shares it with the backend. The backend validates the token and retrieves user details through GoogleAuth. If the token is valid, the backend ensures the user exists in the database, creating a record

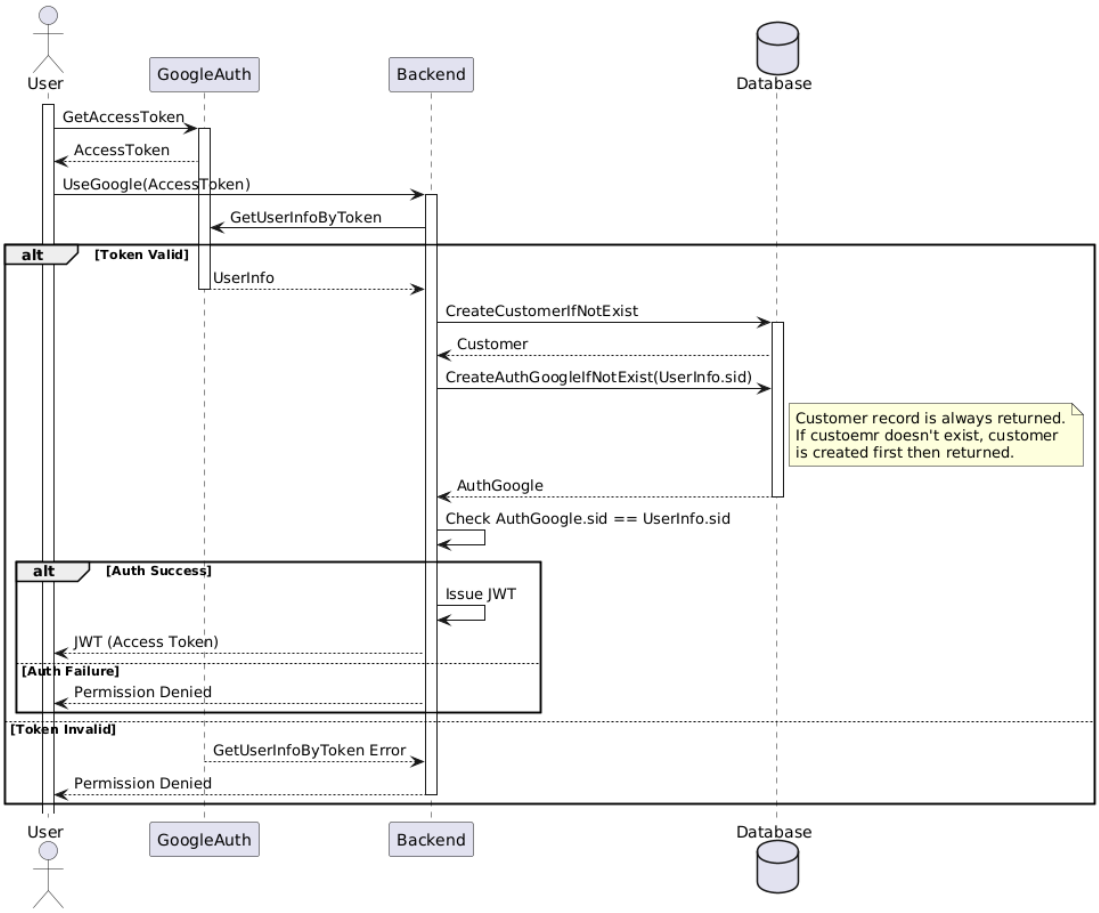

Figure 3.4: Continue with Google Sequence Diagram

if necessary, and associates the user's Google account with it. It then verifies the "sid", issuing a JSON Web Token (JWT) for successful authentication or returning an error, such as Permission Denied, for invalid credentials. This process ensures secure and efficient authentication.

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

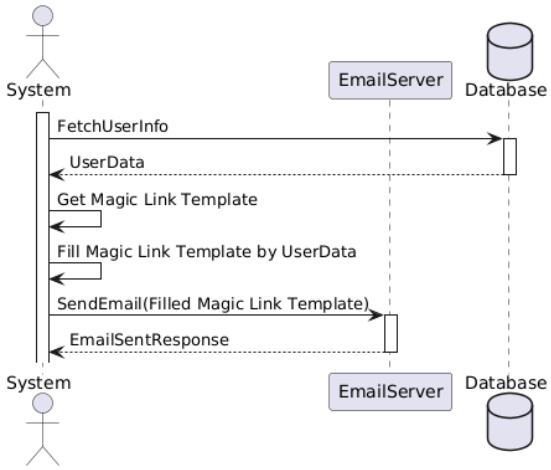

Figure 3.5: Send Welcome Email Sequence Diagram

The "Send a Welcome Email Sequence Diagram", shown in **Figure 3.5**, shows the process of sending a welcome email to new customers. It begins with the system retrieving user data from the database using the FetchUserInfo function. Once the user data is fetched, the system requests a magic link email template via Get Magic Link Template. The template is then customized with the retrieved user data using Fill Magic Link Template by UserData. The system sends the filled email template to the email server via SendEmail, which delivers the email. Finally, the email server responds with EmailSentResponse, confirming the email's successful dispatch. This sequence ensures personalized and reliable email delivery.

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

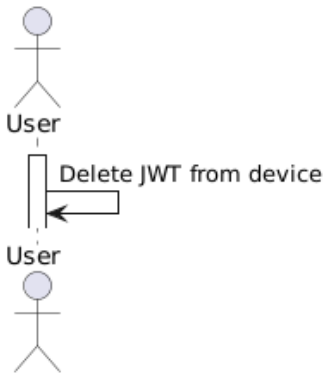

Figure 3.6: Logout Sequence Diagram

The “Logout Sequence Diagram”, shown in **Figure 3.6**, demonstrates the secure logout process in Jadwal. Unlike other operations that require server interaction, the logout process is handled entirely on the client side for efficiency and immediate security effect. When the user initiates logout, the application immediately removes the JWT (JSON Web Token) from the device’s secure storage.

This local-only operation ensures immediate session termination, as any subsequent requests would fail without the authentication token. This approach provides several benefits:

- Instant logout response, regardless of network conditions
- Guaranteed security even if network connectivity is lost
- Clean session termination without server-side state management

After token removal, the user is automatically redirected to the onboarding screen, effectively preventing any further access to authenticated features until a new login is performed.

3.6.2 Calendar Management

At its core, Jadwal helps users manage their calendars effectively. These use cases show how users can create new calendars, connect existing ones through CalDAV, and view all their calendars in one integrated interface.

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

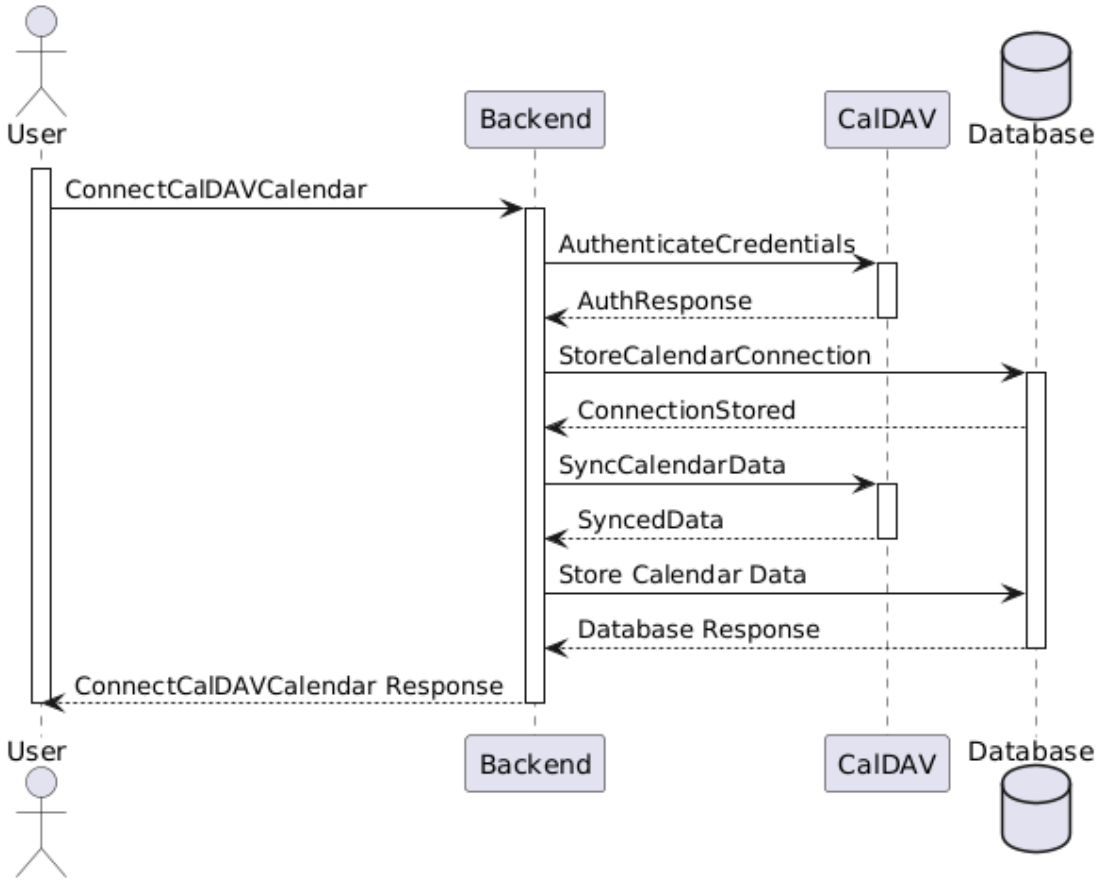

Figure 3.7: Connect Calendar Sequence Diagram

The "Connect Calendar Sequence Diagram", shown in **Figure 3.7**, illustrates the process of connecting an external calendar using CalDAV integration. The sequence begins when the user initiates a calendar connection request through the ConnectCalDAVCalendar gRPC call.

The Backend first authenticates the provided CalDAV credentials with the external calendar service. Upon successful authentication, the Backend securely stores the calendar connection details in the Database, ensuring encrypted storage of sensitive credentials. The Backend then initiates calendar data synchronization, retrieving calendar events and metadata from the CalDAV server.

The synchronized calendar data is stored in the Database, establishing a persistent connection between Jadwal and the user's external calendar. Throughout this pro-

cess, all communication between system components uses gRPC, enabling efficient and type-safe data transfer. The sequence concludes with the Backend sending a ConnectCalDAVCalendar response to the user, confirming successful calendar integration. This workflow ensures secure and reliable calendar connectivity while maintaining data consistency across the system.

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 request 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

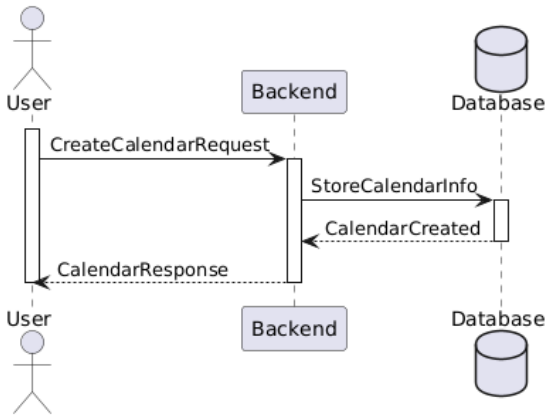

Figure 3.8: Create Calendar Sequence Diagram

The "Create Calendar Sequence Diagram", shown in **Figure 3.8**, illustrates the straightforward process of creating a new calendar within Jadwal. The sequence begins when the user initiates a CreateCalendarRequest gRPC call to the Backend, providing essential calendar details such as name and color preferences.

The Backend processes this request by executing a StoreCalendarInfo operation with the Database, which creates a new calendar record associated with the user's account. Upon successful storage, the Database confirms the creation with a CalendarCreated response. The Backend then sends a CalendarResponse back to the user through the gRPC channel, completing the calendar creation process.

This streamlined sequence ensures efficient calendar creation while maintaining data consistency. The process is designed to be quick and reliable, with proper error handling for network issues or database failures, allowing users to organize their schedules effectively within the Jadwal ecosystem.

Use Case 7: View Integrated Calendar

Basic Information
ID Number: 7 **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	Major Outputs
• **Integrated calendar** (Source: System)	• **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

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 3.7: View Integrated Calendar

The "View Integrated Calendar Sequence Diagram", shown in **Figure 3.9**, demon-

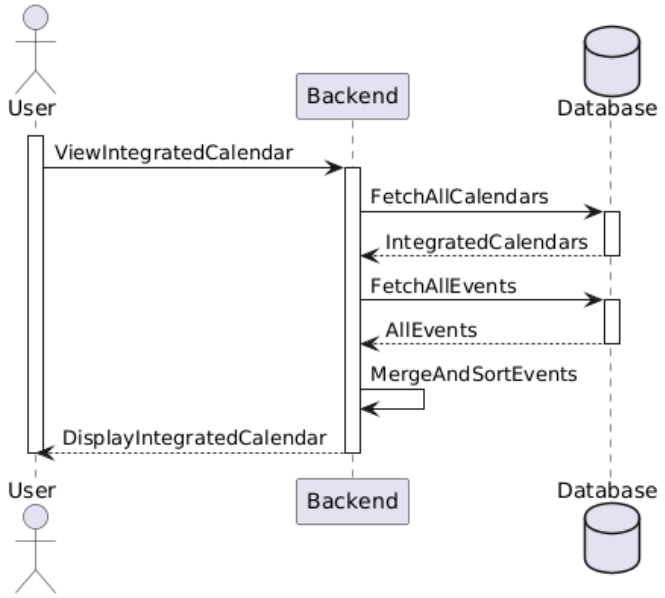

Figure 3.9: View Integrated Calendar Sequence Diagram

strates the process of presenting a unified view of all connected calendars to the user. The sequence begins when the user initiates a ViewIntegratedCalendar gRPC call to access their comprehensive calendar view.

The Backend executes a two-phase data retrieval process:

1. First queries the Database via FetchAllCalendars to retrieve all calendar sources connected to the user's account
2. Then executes FetchAllEvents to gather events from all calendars

After data retrieval, the Backend performs critical processing:

- Merges events from different calendar sources
- Sorts events chronologically
- Applies user preferences for event display

Finally, the Backend returns the processed calendar data through the gRPC channel as DisplayIntegratedCalendar response. This unified view ensures users can efficiently manage their schedule across all connected calendars, maintaining consistency with user preferences while providing a comprehensive overview of all commitments.

3.6.3 WhatsApp Integration and Event Extraction

One of Jadwal's key features is its ability to automatically extract events from WhatsApp conversations. These use cases explain how users connect their WhatsApp account and how the system processes messages to identify and add events to their calendar.

Use Case 8: Connect WhatsApp

Basic Information

ID Number: 8 **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 3.8: Connect WhatsApp

The "Connect WhatsApp Sequence Diagram", shown in **Figure 3.10**, illustrates the two-phase authentication process for connecting a user's WhatsApp account to Jadwal. The sequence begins with the InitiateWhatsApp gRPC call, where the user provides their phone number to the Backend.

The Backend then communicates with the WhatsApp service to request a linking code. This interaction follows two possible paths:

- If the linking code request succeeds:
    1. The user receives the linking code in their WhatsApp application
    2. The user initiates the CompleteWhatsApp gRPC call with the linking code
    3. The Backend validates the code with WhatsApp
    4. Upon successful validation, the WhatsApp authentication credentials are securely stored in the Database for future use
- If the linking code request fails:
    - The Backend immediately returns an InitiateWhatsApp failure response to the user

During the completion phase, if the linking code validation fails, the Backend returns a CompleteWhatsApp failure response, requiring the user to restart the process. This secure two-phase authentication ensures that only legitimate WhatsApp account owners can connect their accounts to Jadwal while maintaining the integrity of the WhatsApp integration.

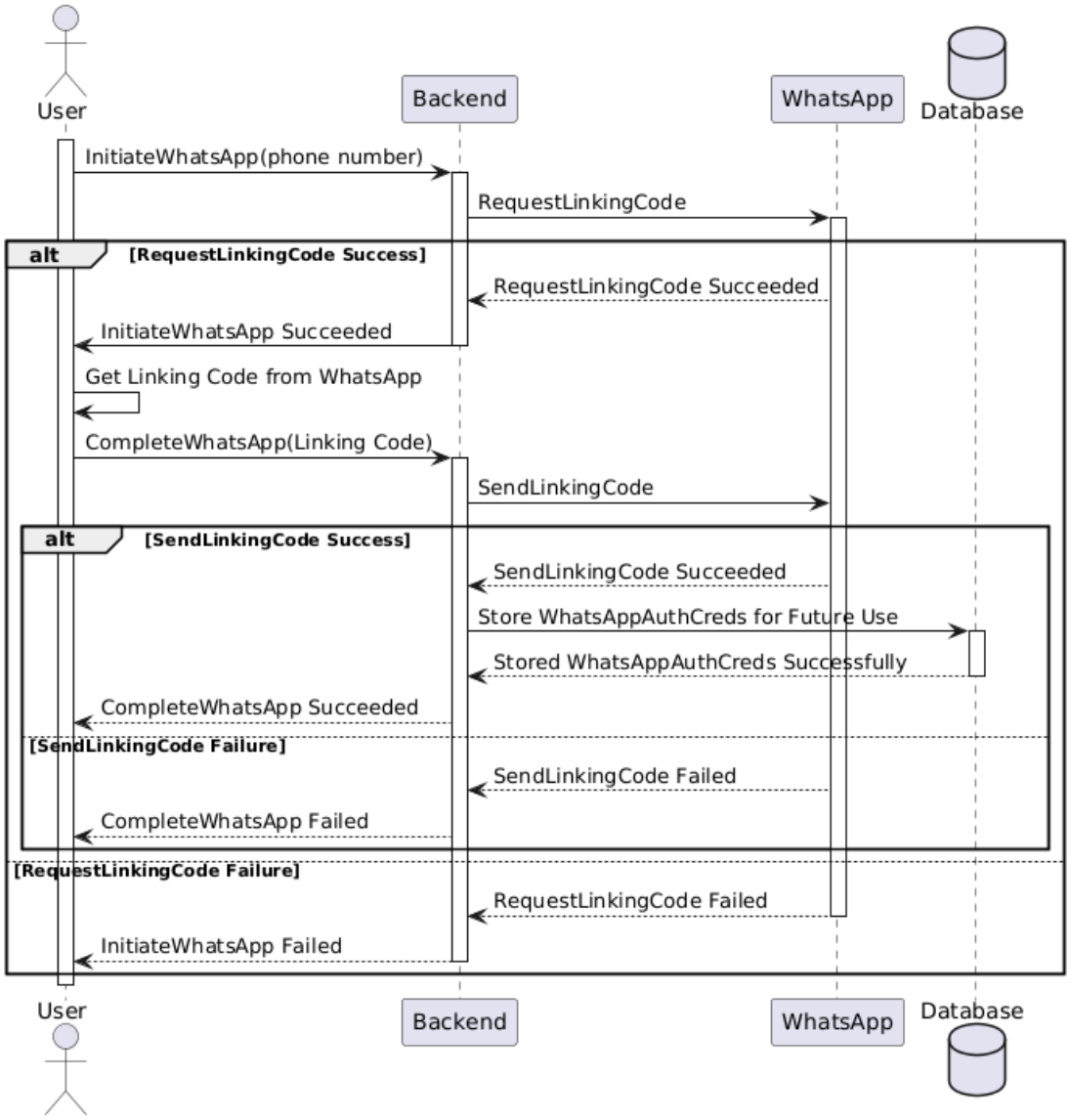

Figure 3.10: Connect WhatsApp Sequence Diagram

Use Case 9: Extract Events from WhatsApp

Basic Information

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

Short Description

System monitors WhatsApp messages of connected WhatsApp accounts and extracts event details using Large Language Models (LLMs), adding them to the user's calendar.

Trigger

A message 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)
- **Context window of last 15 messages** (Source: WhatsApp)

Major Outputs

- **Extracted event details in JSON format** (Destination: System)
- **Push notification to user** (Destination: User's Device)

Main Flow

1. A sent message is received and the user has replied and 30 seconds have passed without any interruptions.

   Information: System reads the last 15 messages to establish conversation context.

2. The conversation context is sent to the LLM service with a carefully engineered prompt.

   Information: The prompt instructs the LLM to analyze messages for event details (date, time, location), consider context, and return structured JSON output while handling various date/time formats.

3. The LLM processes the context and returns event details if found.

   Information: The system validates the extracted information and prepares it for calendar insertion.

4. Add the detected event to the user's calendar.

   Information: System notifies user via push notification 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 ongoing. • The LLM prompt must be engineered to identify potential events, extract key details, and handle various conversation patterns.

Special Requirements
• The system must have access to the user's WhatsApp account as a client to receive and read messages. • The LLM service must be configured to handle natural language processing of informal conversations.

Table 3.9: Extract Events from WhatsApp

The "Extract Events from WhatsApp Sequence Diagram", shown in **Figure 3.11**, illustrates the automated event extraction process from WhatsApp conversations. The system operates in a continuous monitoring loop where the WhatsApp Service (implemented using whatsapp-web.js) listens for incoming messages. Upon receiving a message, the service implements a 30-second delay to gather conversation context, collecting the last 15 messages from the chat history to ensure comprehensive event detection.

The gathered context is then forwarded to the Backend, which interfaces with the LLM Service using a specialized prompt engineered for event detection. This prompt is specifically designed to:

- Identify potential events within informal conversations
- Extract crucial details including date, time, and location
- Process various date and time formats
- Generate structured JSON output for system processing

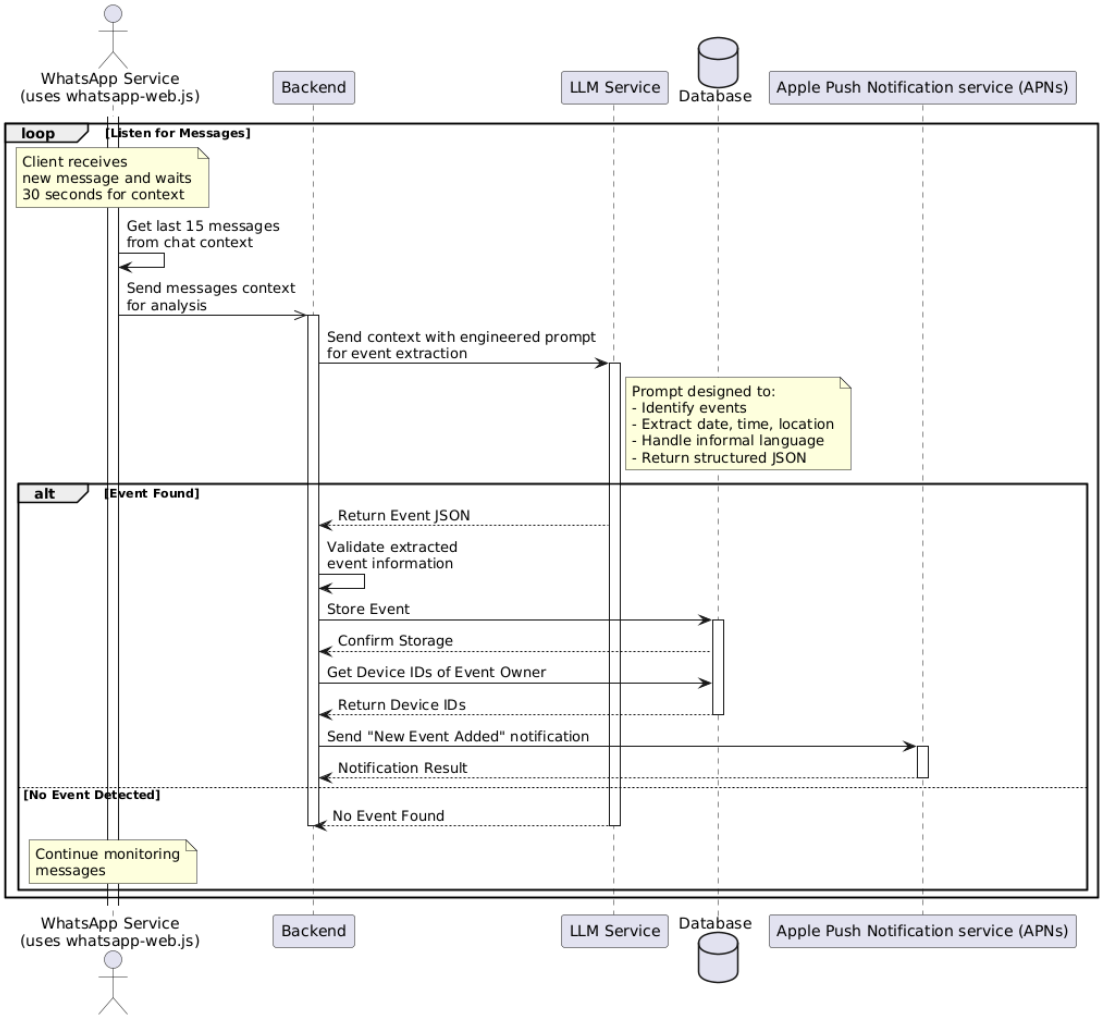

Figure 3.11: Extract Events from WhatsApp Sequence Diagram

When the LLM Service detects an event, it returns a structured JSON response to the Backend. The Backend then:

1. Validates the extracted event information for accuracy
2. Stores the validated event in the Database
3. Retrieves the relevant Device IDs for the event owner
4. Triggers a push notification through Apple's Push Notification service (APNs)

If no event is detected in the analyzed context, the system silently continues its monitoring loop without user notification. This automated workflow ensures efficient event capture from natural conversations while maintaining user awareness through targeted notifications. The system's ability to understand context and process informal language patterns makes it particularly effective for extracting event details from casual WhatsApp conversations.

3.6.4 Event Management and Conflict Resolution

Managing events and resolving scheduling conflicts are daily challenges for users. These use cases demonstrate how Jadwal handles event creation, conflict detection, and provides smart resolution options.

Use Case 10: Suggest Conflict Resolutions

Basic Information

ID Number: 10 **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 3.10: Suggest Conflict Resolutions

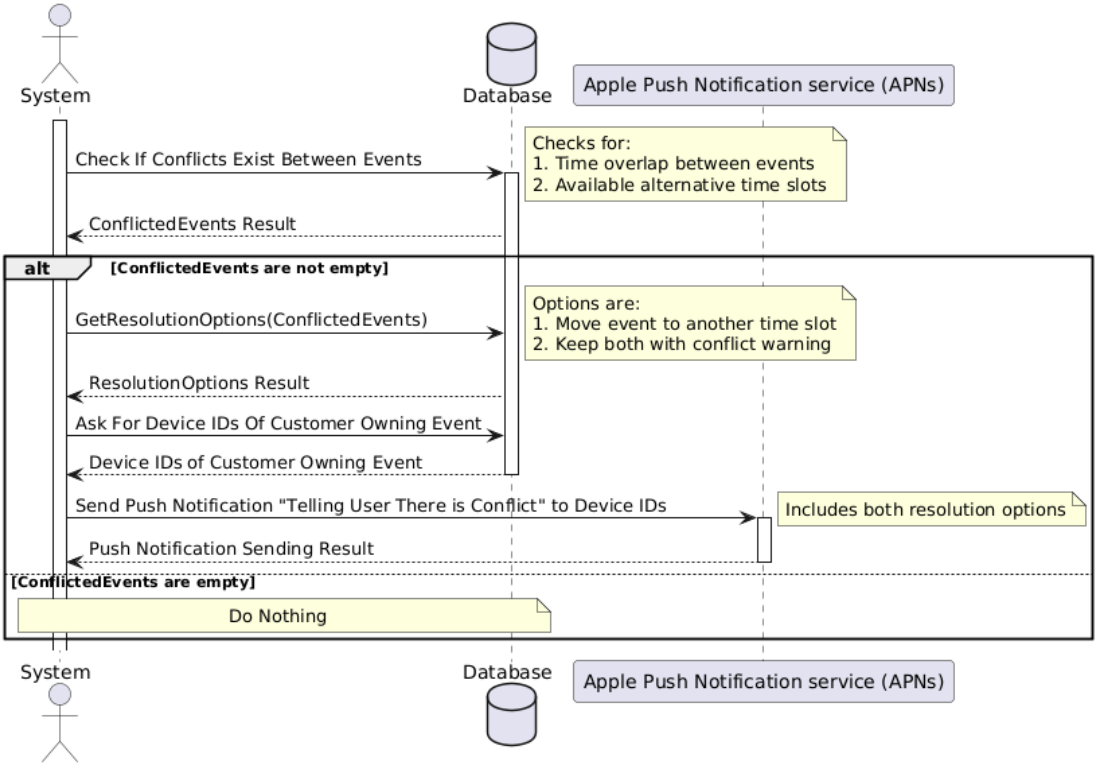

Figure 3.12: Suggest Conflict Resolutions Sequence Diagram

The sequence diagram in Figure 3.12 illustrates the conflict detection and resolution workflow in Jadwal. When events are added to the calendar (either manually or through WhatsApp extraction), the System initiates a conflict check in the Database. This check specifically looks for temporal overlaps between events and identifies potential alternative time slots.

If conflicts are detected, the System follows a structured resolution process:

1. Retrieves resolution options from the Database, which include:
    - Moving the overlapping event to an alternative time slot
    - Keeping both events with an explicit conflict warning
2. Obtains the device IDs associated with the customer who owns the conflicting events
3. Utilizes Apple Push Notification service (APNs) to notify users about the conflict and present them with resolution options

If no conflicts are detected, the System continues without any additional actions. This approach ensures users are promptly informed of scheduling conflicts while maintaining the flexibility to either reschedule events or knowingly maintain overlapping appointments. The workflow aligns with Jadwal's goal of providing straightforward conflict management while respecting user preferences in calendar organization.

This process specifically implements the use case requirements, focusing on practical conflict resolution through user notification and simple resolution options, rather than attempting automated resolution or complex prioritization schemes.

Use Case 11: Manage Scheduling Conflicts

Basic Information
ID Number: 11 **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
• Network 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 3.11: Manage Scheduling Conflicts

The "Manage Scheduling Conflicts Sequence Diagram", shown in **Figure 3.13**, illustrates the interactive process of handling calendar conflicts within Jadwal. The sequence begins with the ViewConflicts gRPC call, where the user requests to see existing scheduling conflicts.

The Backend processes this request through several steps:

1. Retrieves the list of conflicts from the Database via FetchConflicts
2. Generates intelligent resolution options for each conflict
3. Returns the conflicts and their possible resolutions to the user

When the user selects a resolution option, the process continues with:

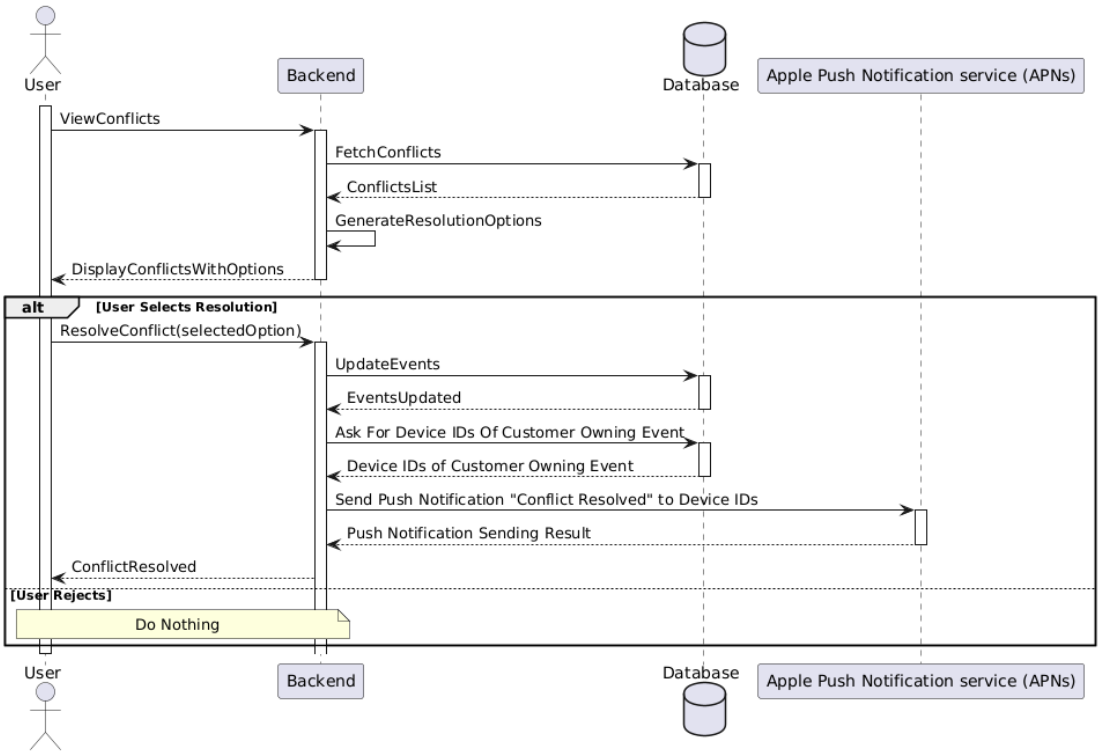

Figure 3.13: Manage Scheduling Conflicts Sequence Diagram

- The ResolveConflict gRPC call carrying the user's selected option
- Database update to reflect the chosen resolution
- Retrieval of relevant device IDs for notification purposes
- Push notification dispatch through Apple Push Notification service (APNs) to inform affected users

If the user chooses to reject resolution, the system maintains the conflict status without further action. This workflow ensures users have complete control over their schedule while maintaining awareness of conflicts through push notifications. The system's ability to generate and present resolution options helps users make informed decisions about their scheduling conflicts.

Use Case 12: Add Event Manually

Basic Information
ID Number: 12 **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 after 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 3.12: Add Event Manually

The "Add Event Manually Sequence Diagram", shown in **Figure 3.14**, illustrates the process flow when a user manually creates a new calendar event. The sequence begins when the user submits event details through the CreateEvent endpoint, triggering a series of validation and storage operations.

The Backend first performs comprehensive validation of the event details, checking for:

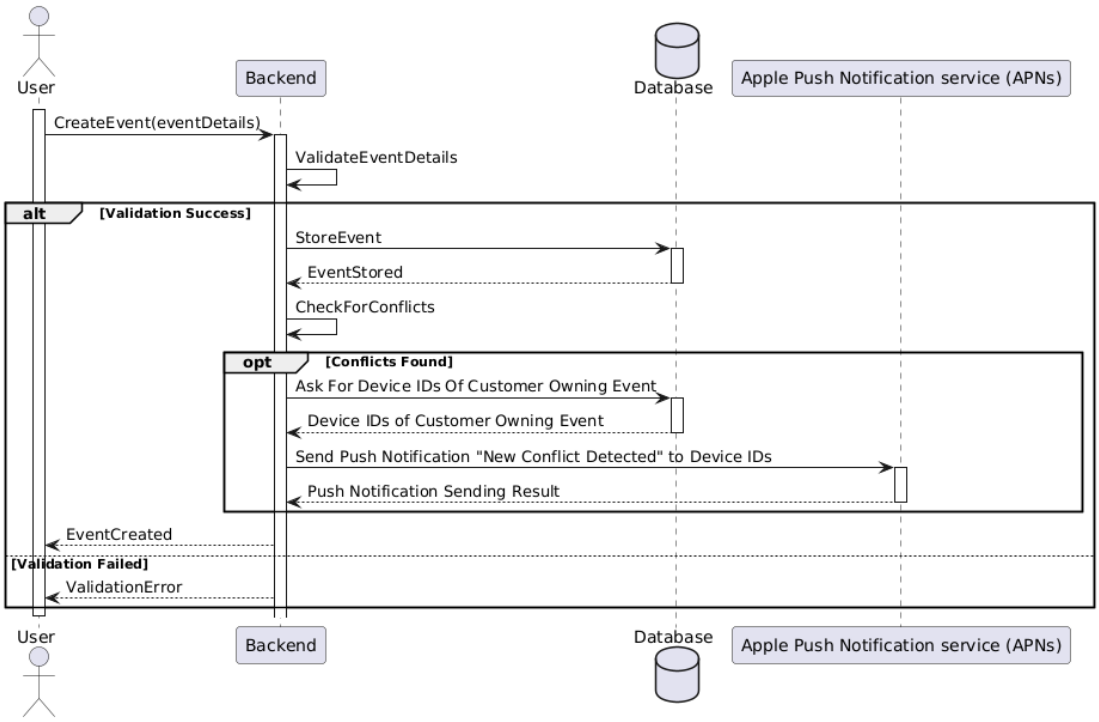

Figure 3.14: Add Event Manually Sequence Diagram

- Mandatory fields completion (event name, date, time)
- Temporal logic (end time after start time)
- Format validity of all provided fields

Upon successful validation, the system executes the following steps:

1. Stores the validated event in the Database
2. Performs an automatic conflict check with existing events
3. If conflicts are detected:
    - Retrieves the device IDs associated with the event owner
    - Dispatches a ”New Conflict Detected” notification via Apple Push Notification service (APNs)
4. Returns an EventCreated response to the user

If validation fails, the system immediately returns a ValidationError to the user, preventing invalid data from entering the system. This workflow ensures data integrity while providing immediate feedback about potential scheduling conflicts, maintaining calendar consistency and user awareness of overlapping appointments.

3.6.5 Prayer Time and Notification Management

Prayer time scheduling is a unique feature of Jadwal, and timely notifications ensure users never miss important events. These use cases detail how prayer times are scheduled and how the notification system keeps users informed.

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	Major Outputs
• **User's time Zone** (Source: user) • **We get the IP address of the user and get their time zone**	• **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

The "Schedule Prayer Times Sequence Diagram", shown in **Figure 3.15**, illustrates

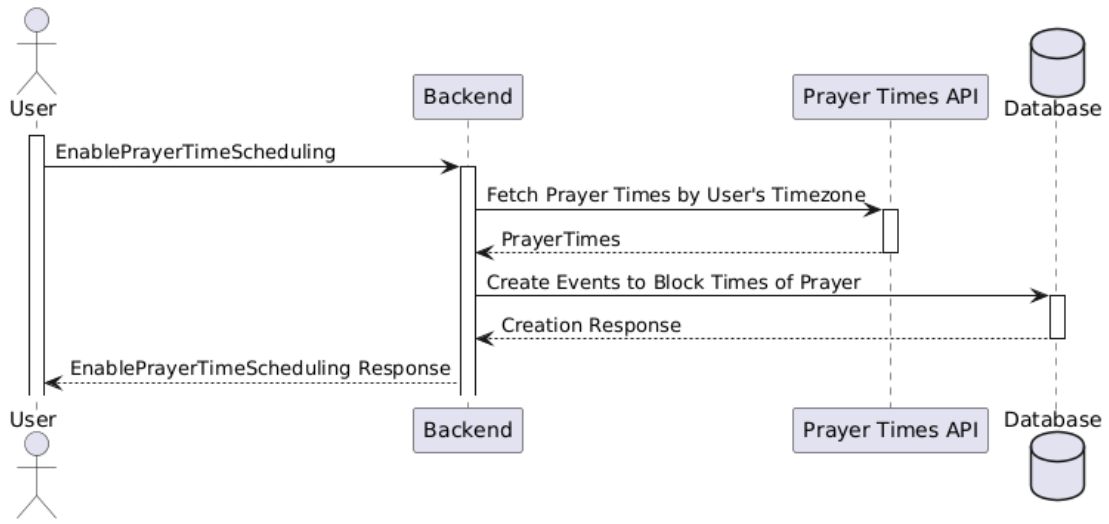

Figure 3.15: Schedule Prayer Times Sequence Diagram

Jadwal's unique feature of automatically scheduling prayer times based on the user's timezone. The process begins when the user initiates the EnablePrayerTimeScheduling gRPC call to activate this feature.

Upon activation, the sequence follows these steps:

1. The Backend determines the user's timezone and queries the Prayer Times API for accurate prayer schedules
2. The Prayer Times API returns the specific times for each prayer based on the user's location
3. The Backend creates blocking events in the Database for each prayer time
4. A confirmation response is sent back to the user through the gRPC channel

This automated process ensures that:

- Prayer times are accurately calculated based on geographical location
- Calendar blocking is automatically adjusted for timezone changes
- Users can maintain their prayer schedule while managing other commitments

The system's ability to automatically block out prayer times demonstrates Jadwal's commitment to supporting users' religious obligations while maintaining efficient schedule management.

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

- **Network 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

The "Receive Event Notifications Sequence Diagram", shown in **Figure 3.16**, illustrates Jadwal's continuous event notification monitoring and delivery system. The sequence operates in a continuous polling loop that executes every minute, ensuring

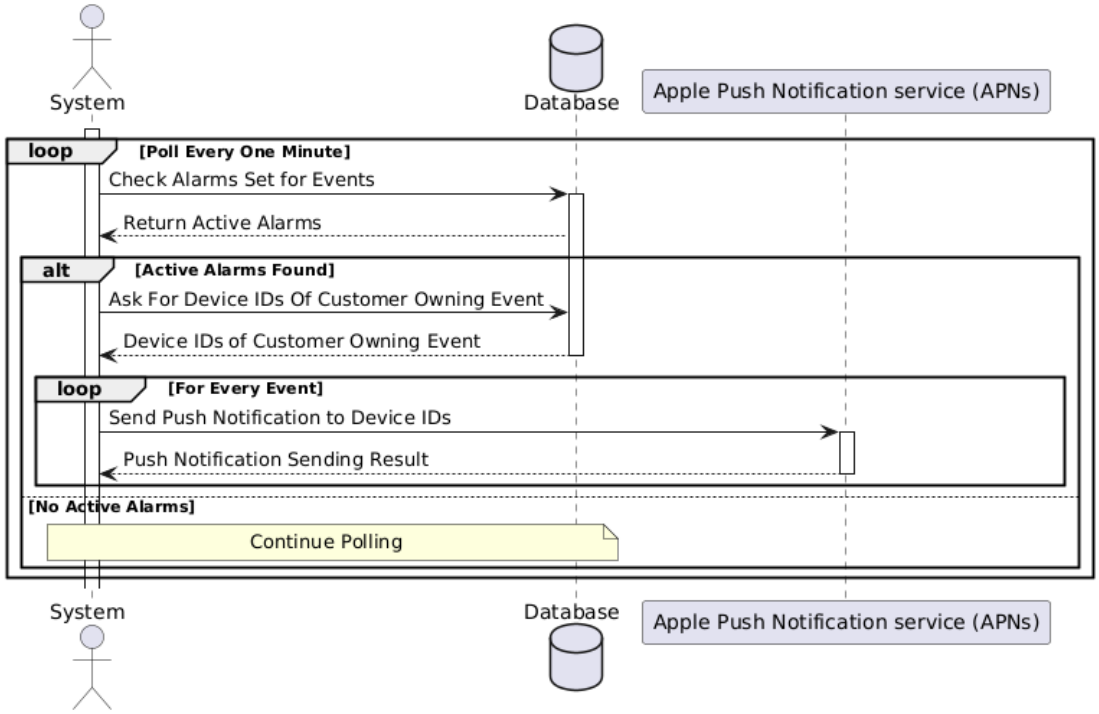

Figure 3.16: Receive Event Notifications Sequence Diagram

timely notification delivery for all scheduled events.

The polling process consists of several key steps:

1. The System queries the Database for active alarms through regular checks
2. For each batch of active alarms found:
    - Retrieves associated device IDs for notification targets
    - Iterates through each event requiring notification
    - Dispatches push notifications via Apple Push Notification service (APNs)
3. If no active alarms are found, the system continues its polling cycle

This robust notification system ensures reliable delivery of event reminders while efficiently managing system resources. The one-minute polling interval provides a balance between timely notifications and system performance, while the batch processing of notifications optimizes the interaction with APNs. The system's ability to handle multiple device IDs per user ensures notifications reach users across all their registered devices.

3.7 Activity Diagram

Activity diagrams provide a visual representation of the workflow and business processes within a system. For Jadwal, the activity diagram maps out the user's journey through the application, from authentication to daily interactions, illustrating the various paths and decision points users encounter. This diagram is particularly important as it demonstrates how different features of Jadwal—such as calendar management, WhatsApp integration, and conflict resolution—interconnect in actual usage.

An activity diagram is shown in Figure 3.17 above, and it illustrates the flows the user of the app can take. It starts with the authentication which supports both Google and Email.

Then after 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 the previous 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.

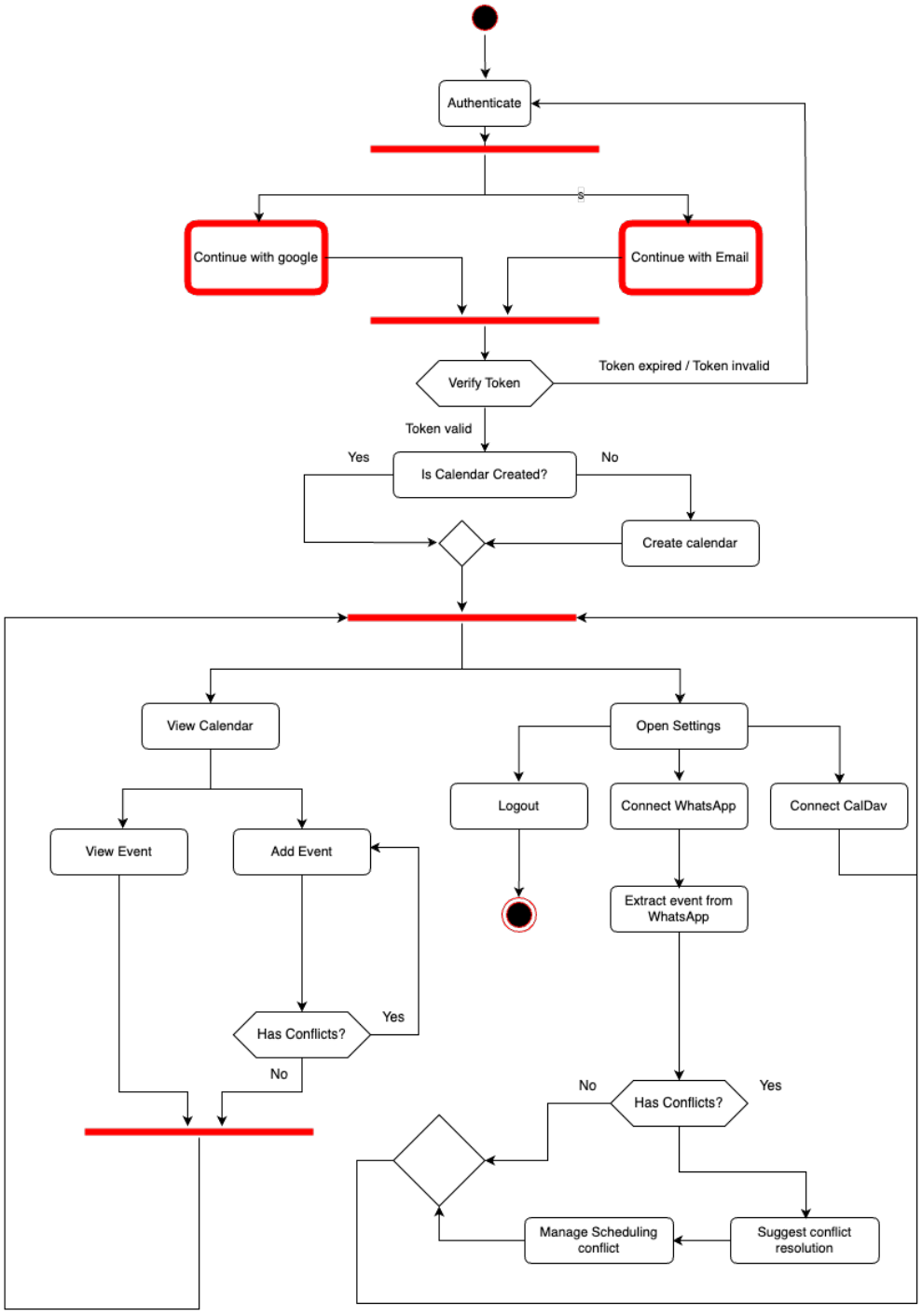

Figure 3.17: Activity Diagram of Jadwal

3.8 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)

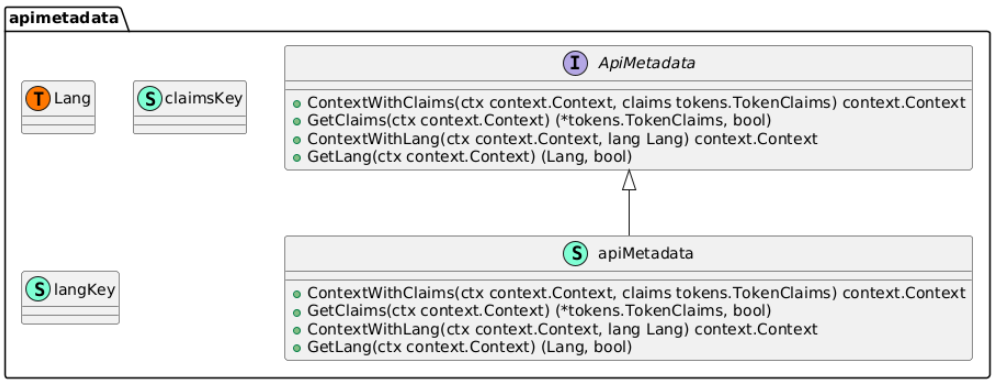

Figure 3.18: Api Metadata Class Diagram

In Figure 3.18, 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.

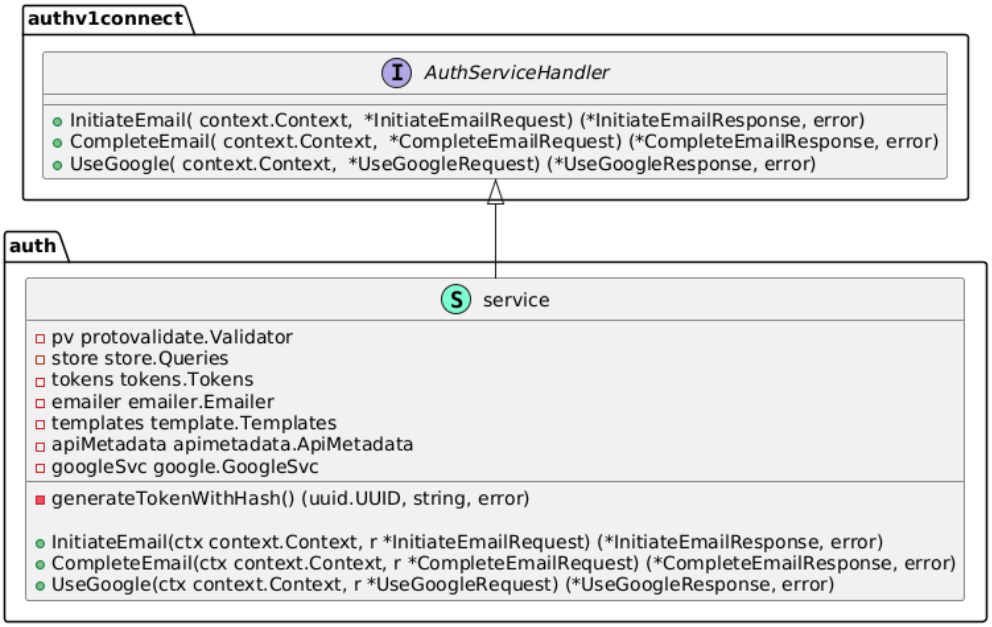

Figure 3.19: Auth Class Diagram

In Figure 3.19, 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.

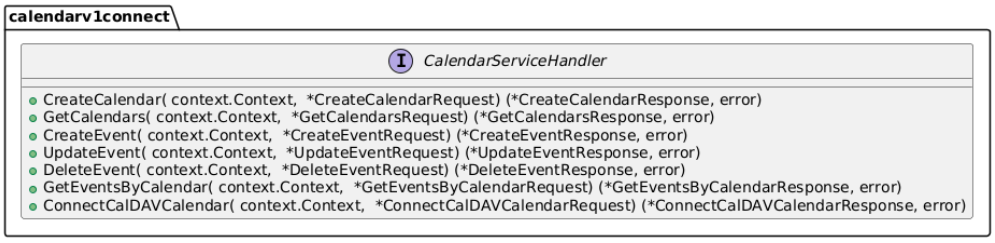

Figure 3.20: Calendar V1 Class Diagram

In Figure 3.20, 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.

In Figure 3.21, 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

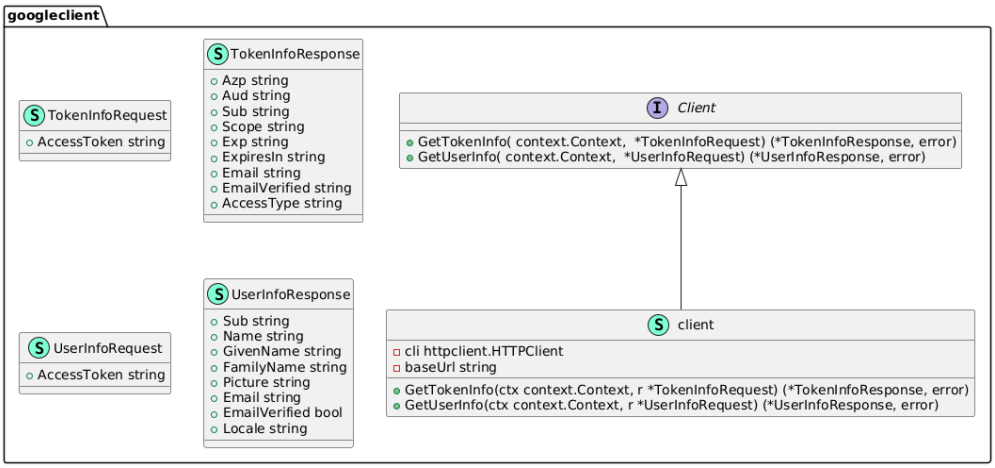

Figure 3.21: Google Client Class Diagram

info the system was granted access for.

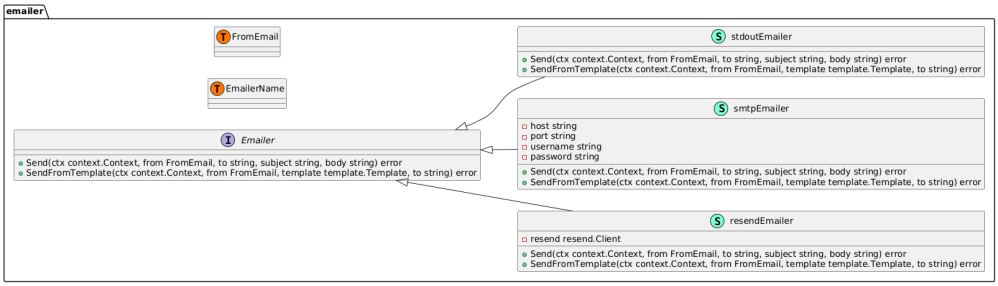

Figure 3.22: Emailer Class Diagram

In Figure 3.22, 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.

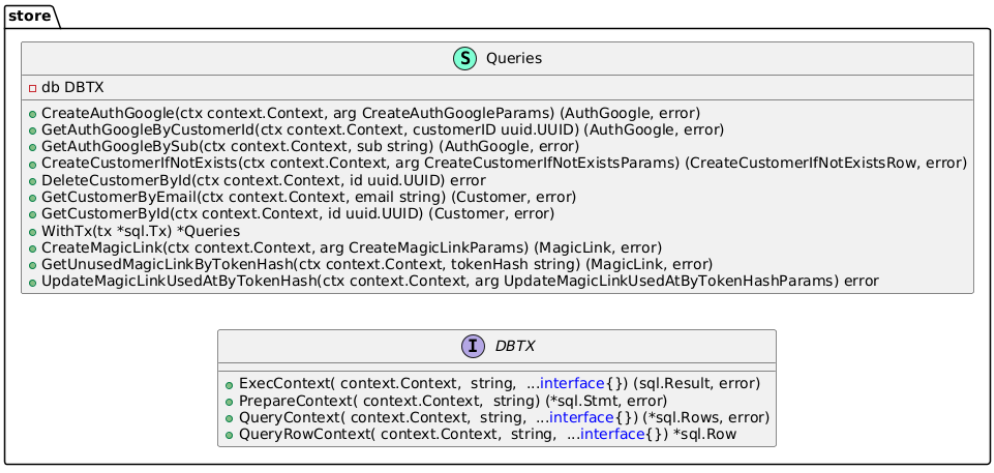

Figure 3.23: Store Class Diagram

In Figure 3.23, 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.

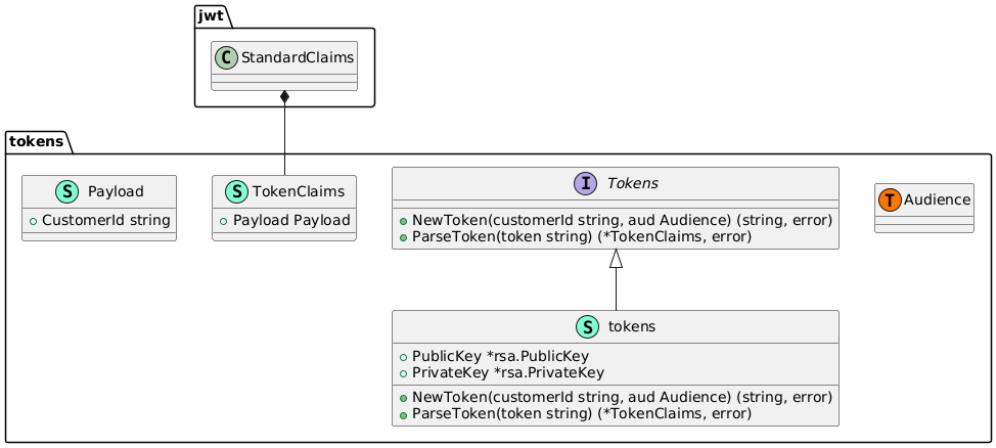

Figure 3.24: Tokens Class Diagram

In Figure 3.24, 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.

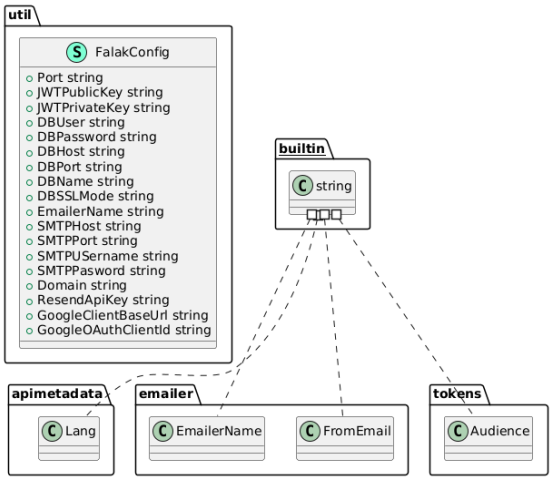

Figure 3.25: Util Class Diagram

In Figure 3.25, 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.

3.9 Database Design

The foundation of Jadwal's robust functionality lies in its carefully structured relational database architecture. This section presents both the Entity-Relationship model and Relational Schema that support the application's core features, from user authentication to calendar integration and event management. The database design ensures efficient data organization while maintaining the flexibility needed for future scalability.

3.9.1 Entity-Relationship Model

The Entity-Relationship model, illustrated in **Figure 3.26**, depicts the conceptual structure of Jadwal's database system. This model follows standard ER notation and demonstrates the relationships between the system's primary entities.

Core Entities and Relationships

1. **User Management**
    - `customer`: Central entity with attributes `id` (UUID), `name`, and `email`
    - `auth_google`: Represents Google OAuth authentication
    - `magic_link`: Manages email-based authentication tokens
2. **Calendar Organization**
    - `calendar_accounts`: Links customers to calendar providers
    - `vcalendar`: Represents individual calendars with metadata
    - One-to-many relationship from accounts to calendars
3. **Event Management**
    - `vevent`: Core event entity with timing and details
    - `vevent_exception`: Handles recurring event modifications
    - `valarm`: Manages event notifications

Key Relationships

- Customer ”has” authentication methods (one-to-many)
- Customer ”owns” calendar accounts (one-to-many)
- Calendar accounts ”contain” calendars (one-to-many)
- Calendars ”schedule” events (one-to-many)
- Events ”have” exceptions and alarms (one-to-many)

3.9.2 Relational Database Schema

The relational database schema, shown in **Figure 3.27**, provides the detailed logical structure of the database implementation. This representation demonstrates the actual table structures, attributes, and relationships as implemented in the database system.

Table Structures

1. **Authentication Tables**
   - `customer` (id, name, email, created_at, updated_at)
   - `auth_google` (id, customer_id, sub, created_at, updated_at)
   - `magic_link` (id, customer_id, token_hash, expires_at, used_at, created_at, updated_at)
2. **Calendar Tables**
   - `calendar_accounts` (id, customer_id, provider, created_at, updated_at)
   - `vcalendar` (uid, account_id, prodid, version, display_name, description, color, timezone, created_at, updated_at)
3. **Event Tables**
   - `vevent` (uid, calendar_uid, dtstamp, dtstart, dtend, duration, summary, location, status, classification, transp, rrule, rdate, exdate, sequence, created_at, updated_at)

- `vevent_exception` (id, event_uid, recurrence_id, summary, description, location, dtstart, dtend, status, created_at, updated_at)
- `valarm` (id, event_uid, action, trigger, description, summary, duration, repeat, attendees, created_at, updated_at)

Implementation Details

- **Primary Keys**: Underlined attributes represent primary keys
- **Foreign Keys**: Attributes ending in "_id" or "_uid" represent foreign key relationships
- **Audit Fields**: All tables include created_at and updated_at timestamps
- **Data Types**:
  - UUID for unique identifiers
  - VARCHAR for variable-length strings
  - TIMESTAMPTZ for timezone-aware timestamps
  - JSONB for complex data structures (rdate, exdate, attendees)

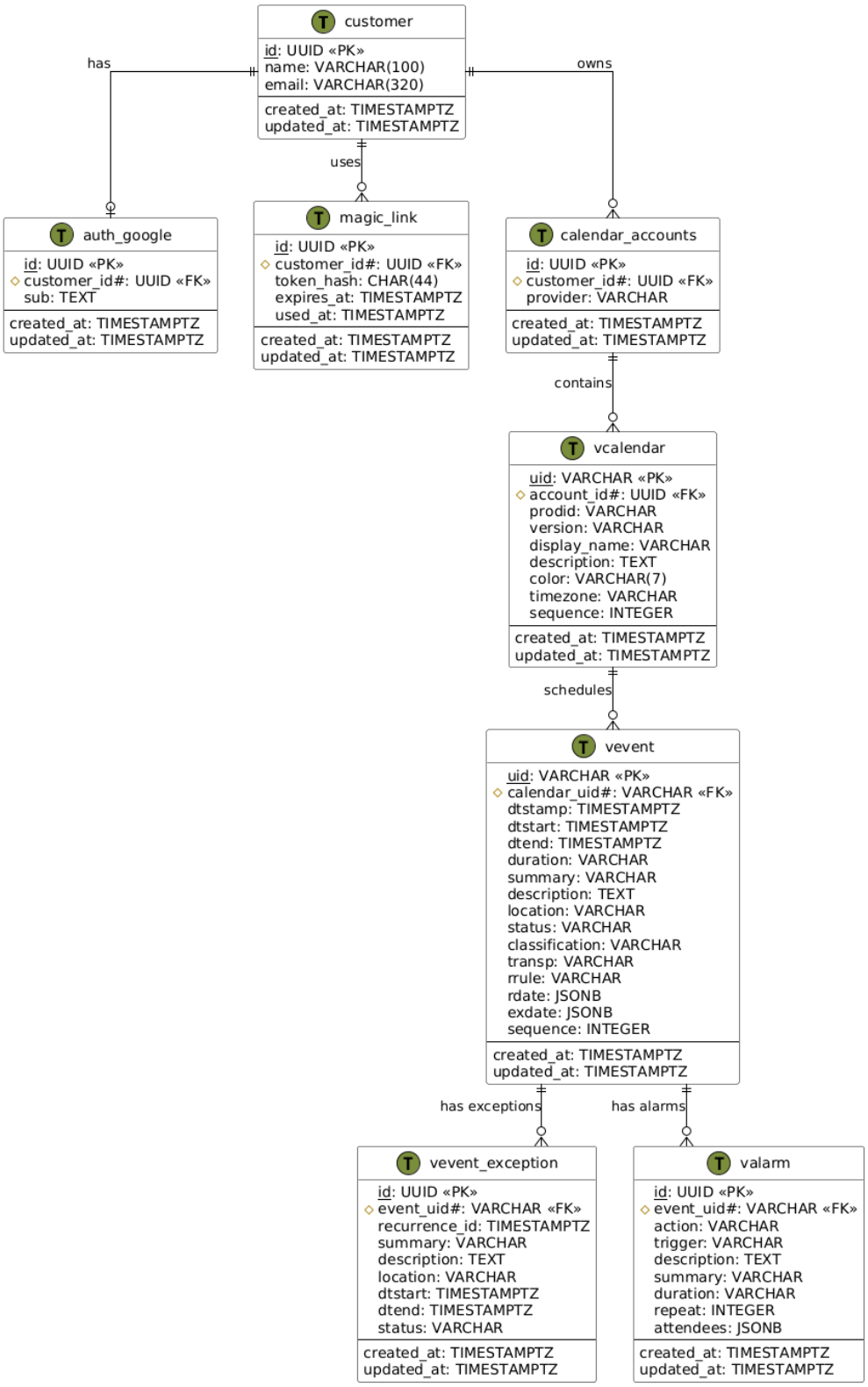

Figure 3.26: Entity-Relationship Diagram

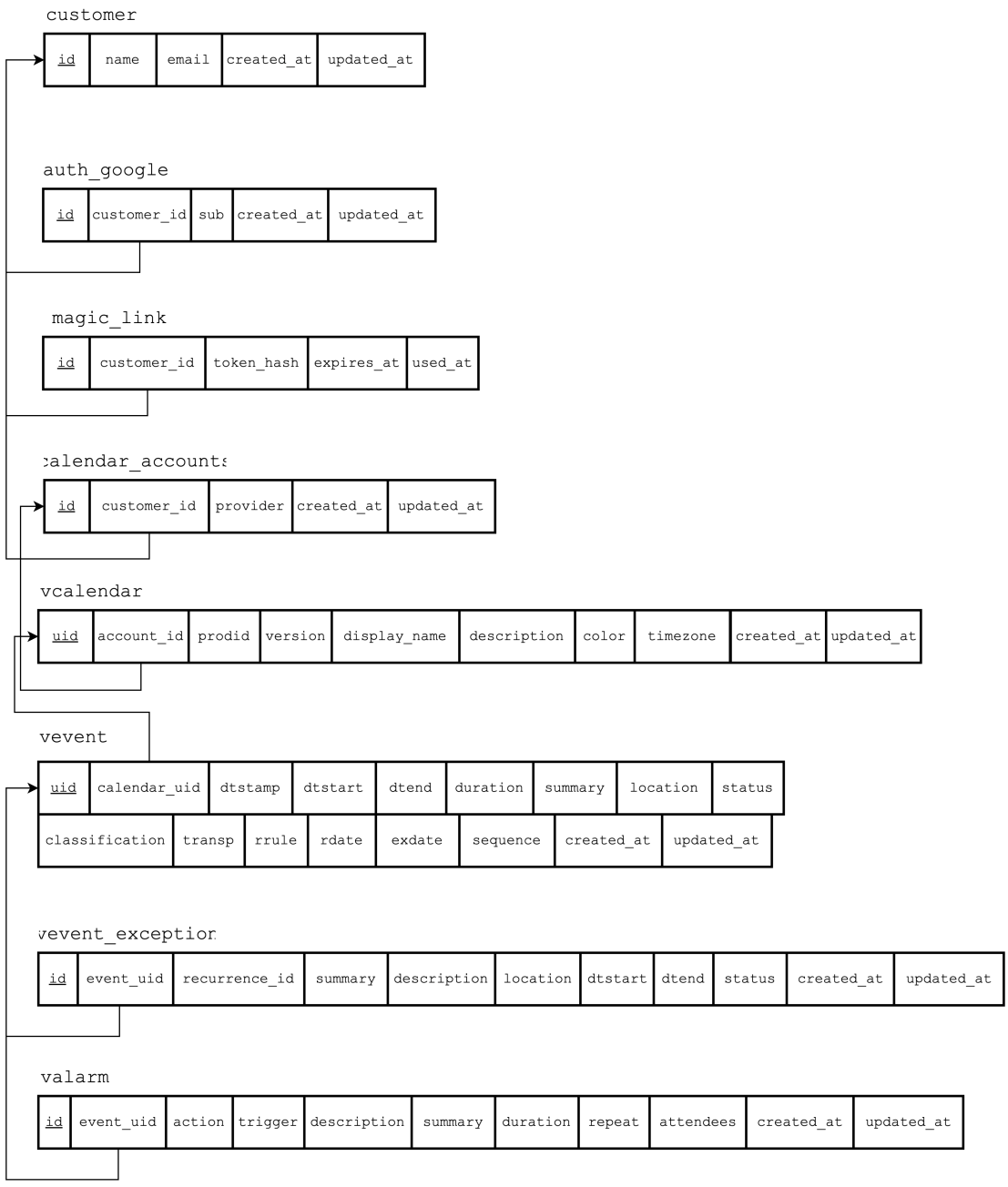

Figure 3.27: Relational Schema

3.10 User Interface Prototype

The user interface design of Jadwal plays a crucial role in delivering an intuitive and efficient calendar management experience. This section presents a comprehensive walkthrough of the application's key screens, from initial user onboarding to daily calendar interactions. The prototype demonstrates how Jadwal's core features—including authentication, calendar viewing, event management, and settings configuration—are implemented in a user-friendly manner that adheres to iOS design principles. Each screen has been carefully designed to balance functionality with simplicity, ensuring users can easily navigate and utilize all of Jadwal's features.

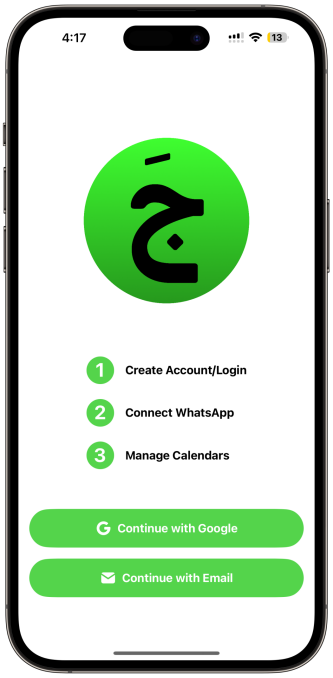

Figure 3.28: UI Screen 1: Onboarding View

In Figure 3.28, 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 3.29, 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.

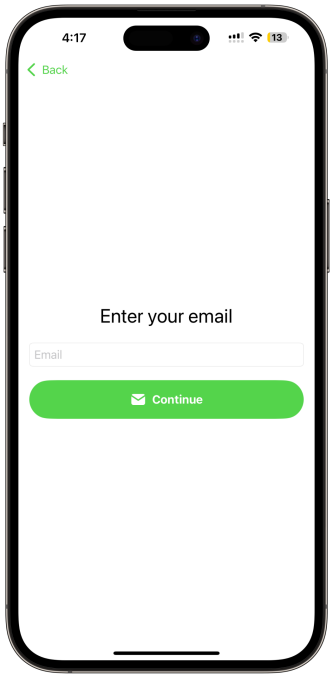

Figure 3.29: UI Screen 2: Continue with Email View

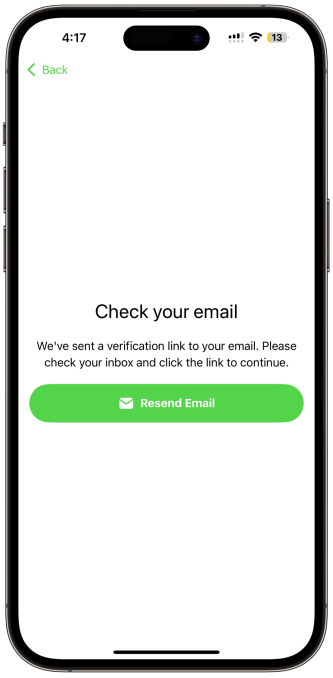

Figure 3.30: UI Screen 3: Check Your Email View

In Figure 3.30, 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 3.31

In Figure 3.31, 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 3.31: UI Screen 4: Calendar View

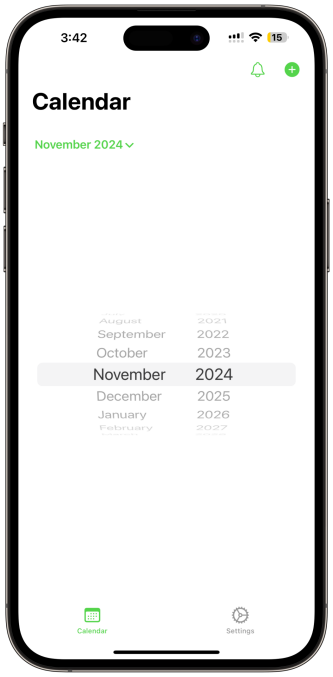

Figure 3.32: UI Screen 5: Month & Year Selector

In Figure 3.32, 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 3.33, the screen is showing the add event sheet in its default state. It is shown when you click the ”add event” button in Figure 3.31.

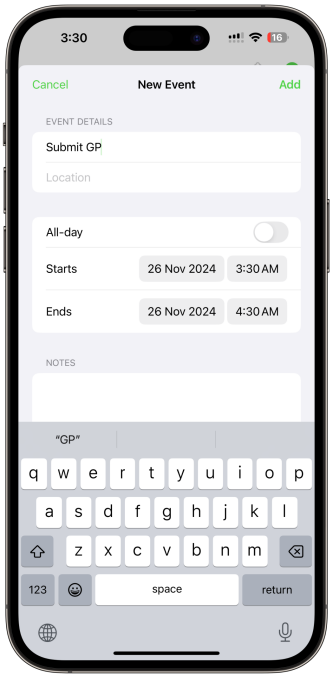

Figure 3.33: UI Screen 6: Add Event View - Default

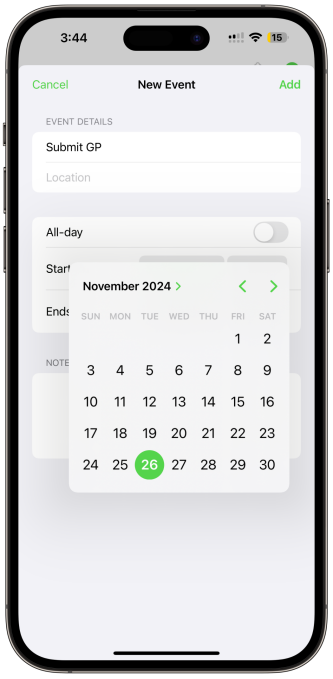

Figure 3.34: UI Screen 7: Add Event View - Date Picker

In Figure 3.34, 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 3.35, 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.

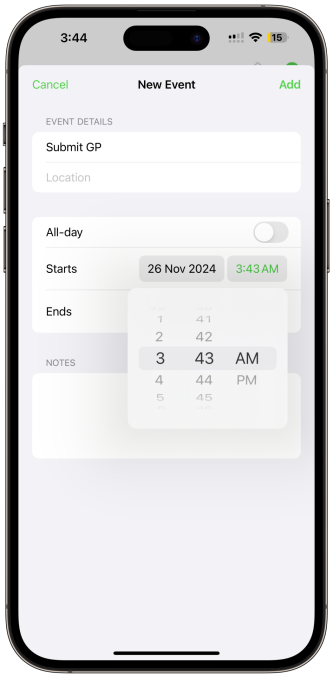

Figure 3.35: UI Screen 8: Add Event View - Time Picker

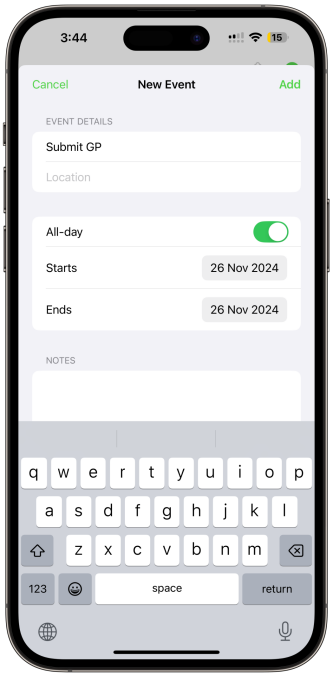

Figure 3.36: UI Screen 9: Add Event View - All Day

In Figure 3.36, 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 3.37, the settings page is shown. This page has the user details, specifically 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.

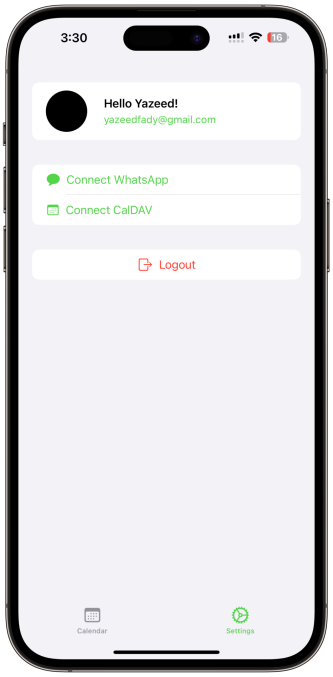

Figure 3.37: UI Screen 10: Settings View

3.11 Conclusion

Jadwal stands as a comprehensive and innovative solution for scheduling with great design. The use cases and database architecture are well described, which 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.

CONCLUSION

In a fast-paced and interconnected world, everyone struggles to manage their time efficiently, especially for someone who has multiple tasks every day. Jadwal addresses all these challenges by introducing features that help to bridge all the gaps that other platforms are unable to offer. Jadwal ensures a user-friendly interface for the user to manage their schedule and avoid missing any events.

Jadwal integrates with CalDAV, a standardized protocol that facilitates seamless synchronization across various calendar platforms. This allows users to effortlessly connect and manage their calendars from different devices and applications, such as Apple Calendar and Google Calendar, making everything in one ecosystem.

Jadwal's state-of-the-art LLM WhatsApp event extraction feature enhances the application by extracting events from WhatsApp, which is an informal communication channel that people most likely use for discussing events. This creates a gap that could lead to missing or conflicting events, and Jadwal aims to solve this problem.

The feature of scheduling prayer times highlights Jadwal as unique, and Jadwal sees it as important since it is a daily activity. Clearing and holding a place for this event would help the user to manage their day more efficiently.

The feature that makes Jadwal stand out is managing events effectively by offering features like suggesting conflict resolution and integrating other calendars. This would reduce the chances of the user missing any events.

Integration with CalDAV allows Jadwal to connect all the calendars that the user has and bring them to one platform, which helps a lot in managing the day. Why do we separate calendars to organize one day for one person?

Throughout the project, a systematic approach was taken in developing Jadwal. 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 architecture and system design showcase the backbone of Jadwal, illustrating how the system works and how the data is managed perfectly. These technical aspects ensure that Jadwal is well-structured and easy to understand, making it simple to track everything happening in the backend.

In conclusion, Jadwal stands out from all the existing solutions and provides an innovative calendar application that bridges the gaps and helps the user to manage their day more efficiently. With its comprehensive features, user-friendly interface, and robust technical foundation, Jadwal is poised to revolutionize the way individuals manage their time in today's fast-paced world.

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.

[gRPC, 2024] gRPC (2024). grpc definition. Accessed: 2024-12-02.

[HTTP2, 2024] HTTP2 (2024). Http/2 definition. Accessed: 2024-12-02.

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

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

[NodeJS, 2024] NodeJS (2024). Node.js definition. Accessed: 2024-12-02.

[PostgreSQL, 2024] PostgreSQL (2024). Postgresql definition. Accessed: 2024-12-02.

[Protobuf, 2024] Protobuf (2024). Protocol buffers definition. Accessed: 2024-12-02.

[RabbitMQ, 2024] RabbitMQ (2024). Rabbitmq definition. Accessed: 2024-12-02.

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

[SwiftUI, 2024] SwiftUI (2024). Swiftui definition. Accessed: 2024-12-02.

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

[WebDAV, 2007] WebDAV (2007). Webdav definition. Accessed: 2024-11-27.

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

[WhatsAppWebJS, 2024] WhatsAppWebJS (2024). Whatsapp web.js definition. Accessed: 2024-12-02.