COMP442 – Software Engineering Lab

Schedule Assistant Bot

Rem

Group 2  
Hadi Salloum 201900774  
Khaled Sardouk 201901490  
Mohammad Al Tayyeb Soubra 201901076  
Mohamad Alaili 201901052

**Contents**

[I. Project Description 3](#_Toc72727221)

[1. Project Overview 3](#_Toc72727222)

[2. The Purpose of The Project 3](#_Toc72727223)

[3. The Scope of The Work 3](#_Toc72727224)

[4. Product Scenarios 4](#_Toc72727225)

[5. Stakeholders 4](#_Toc72727226)

[6. Mandated Constraints 4](#_Toc72727227)

[7. Naming Conventions and Definitions 4](#_Toc72727228)

[8. Relevant Facts and Assumptions 4](#_Toc72727229)

[II. Model Used 5](#_Toc72727230)

[III. Requirements: 5](#_Toc72727231)

[1. Functional Requirements: 5](#_Toc72727232)

[A. User 5](#_Toc72727233)

[B. System 5](#_Toc72727234)

[2. Non-Functional Requirements: 5](#_Toc72727235)

[A. Performance Requirements 5](#_Toc72727236)

[B. Dependability Requirements 6](#_Toc72727237)

[C. Maintainability and Supportability Requirements 6](#_Toc72727238)

[D. Security Requirements 6](#_Toc72727239)

[E. Usability and Humanity Requirements 7](#_Toc72727240)

[F. Look and Feel Requirements 7](#_Toc72727241)

[G. Operational and Environmental Requirements 7](#_Toc72727242)

[H. Cultural and Political Requirements 7](#_Toc72727243)

[I. Legal Requirements 7](#_Toc72727244)

[IV. General Design: Bot Commands 9](#_Toc72727245)

[V. UML Design: 11](#_Toc72727246)

[1. UML Use Case Design: 11](#_Toc72727247)

[2. UML Class Model 12](#_Toc72727248)

[3. UML Sequence Diagram 13](#_Toc72727249)

[4. UML State Machine 14](#_Toc72727250)

# Project Description

## Project Overview

“Schedule Bot” is a software that’s designed to help school and studying group chats to keep track of important due dates like assignment and exam deadlines in popular messaging apps like Telegram and Discord.

## The Purpose of The Project

Remind users of important deadlines so it makes it easier for users to properly prioritize tasks by the date they’re due than if they had to go to repeatedly each separate course page and compare deadlines manually.

## The Scope of The Work

* **Milestones:**

In phase 1, we will focus on getting the basic functionality that is reminding participants of group chats in Telegram of deadlines accurately. Hence, we will be developing a python bot that interacts with the Telegram API.

In phase 2, we will expand out bot to also work with Discord API so users of servers in Discord can also benefit from this bot.

In phase 3, which we may not reach depending on time, we will try and develop a standalone application that provides similar functionality and can serve as an administerial hub for users to modify their own deadlines in the group chat.

* **Deliverables:**

We will a deliver a bot that satisfies the requirements we set and serves as an assistant for group chat. If we manage to reach the **third phase**, wewill also deliver an android application that offers additional functionality.

We will also provide biweekly reports that track our progress through development.

* **Timeline:**

Expected Duration: 3 months

Phase 1: 01-March => 30-April

Phase 2: 01-May => 14-May

Phase 3: 14-May => 31-May

* **Reports:**

Biweekly Reports should be written concerning the progress of the project. They should include progress towards milestones and notable accomplishments during those 2 weeks. Also, they should include any issues or setbacks faced and how they were handled or to be handled in the future.

* **Budget:**

Python and Android Studio available for free.

Telegram and Discord API necessary to have the bot communicate with them is also available for free.

## Product Scenarios

* During initialization, an authorized user enters future deadlines and reminders that the bot should be responsible for.
* Any other user in the group chat asks the bots for the deadlines ordered by closest date.
* The bot provides the deadlines.

## Stakeholders

* Group Chats: have the bot remind the whole group chat of deadlines set by authorized members of the group chat.
* Normal Users that could be members of study groups or any other group that needs the functionality our software provides:
* Authorized Group Members: Assign deadlines and due dates for the bot to keep track of for the whole group chat.

## Mandated Constraints

* The software should not report deadline inaccurately.
* The software should handle errors such as invalid dates properly.
* Modifying deadlines should only be allowed to authorized users of group chats.

## Naming Conventions and Definitions

Discord is an American VoIP, instant messaging and digital distribution platform designed for creating communities. Users communicate with voice calls, video calls, text messaging, media, and files in private chats or as part of communities called "servers".[1]

Telegram is a freeware, cross-platform, cloud-based instant messaging software and application service. The service also provides end-to-end encrypted video calling, VoIP, file sharing and several other features.[2]

## Relevant Facts and Assumptions

* The bot will be on available on telegram so on any platform that supports Telegram such as iOS, android, Windows OS (Telegram Web)… etc.

[1] <https://en.wikipedia.org/wiki/Discord_(software)>

[2] <https://en.wikipedia.org/wiki/Telegram_(software)>

# Model Used

The SDLC used is the waterfall model. We used this model since our project is small sized one and its implementation is straightforward. It’s neither large nor complex. The other model we considered was the Kanban model, but since the Kanban model relies a bit on feedback from the customers, and we don’t really need customer feedback in our developing stages, only after we are done implementation. Therefore, we will use the waterfall model.

# Requirements:

## Functional Requirements:

### User

* Users should be able to invite the bot to their group chats.
* Authorized users should be able to add, delete, or modify reminders.
* All users should be able to view reminders
* Users should be notified concerning certain remainders depending on the parameters set for each reminder.
* Users shall be able to request the reminders be sorted by date due, priority, type… etc.

### System

* The system should not confuse the reminders of a group chat with those of another group chat.
* System should be able to receive and perceive dates accurately, for example: the system should be able to tell the difference between daylight saving time and standard time.
* The system should support different time zones and converting reminders deadlines to the time zone of the user requesting.
* The system should delete reminders properly when they expire.
* The system should never forget reminders.
* The system should be able to receive various data about each reminder like the type of the reminder, its priority, and be able to send this data back to users when requested

## Non-Functional Requirements:

### Performance Requirements

Speed and Latency Requirements: Any interface between the software and the user should have a maximum response time of 1 second.

Precision or accuracy Requirements: All reminders should be sent at the accurate and correct times. Moreover, all the data sent from the software to the user should be correct and properly represent the entered/inputted data into the software.

Capacity Requirements: The software should be able to keep tracks of a large number of schedules at the same time for one individual and should also be able to interact with a very large number of group chats simultaneously.

### Dependability Requirements

Reliability Requirements: The system should almost never fail and in case it does, it should aim to resolve such failures smoothly.

Availability Requirements: The service should be available for use 24 hours per day, 365 days per year achieving about 99% uptime.

Fault-tolerance Requirements: The software should preserve all data associated with its user even if not all user are accessible by storing the data on a central server and maintaining the connection to it.

### Maintainability and Supportability Requirements

Maintenance Requirements: The ability of fixing code bugs easily without the need to alter the whole code. The code should be implemented using classes and methods for ease of reuse.

Availability Requirements: Any user using a modern device capable of utilizing Telegram (or Discord) should be able to interface with the product.

Scalability or Extensibility Requirements: The product should be capable of interacting with an unlimited number of users. The product should be able to send 10,000 schedule reminders per hour within two years of its launch.

Longevity Requirements: The product should be expected to run within a maximum maintenance budget for at least 4 years.

### Security Requirements

Access Requirements: Only users of the group chat that added the schedules should be able to see and modify them. Only the developers of the application should be aware of the group chats that the schedule bot is a part of.

Integrity Requirements: The product should prevent incorrect data from being introduced and it should protect itself from intentional abuse.

Privacy Requirements: The product should make its users aware of its information practices before collecting data from them. The product should notify customers of changes to its information policy. The product should reveal private information only in compliance with the organization’s information policy. The product should protect private information in accordance with the relevant privacy laws.

Immunity Requirements: The product should protect itself from unauthorized usage.

### Usability and Humanity Requirements

Ease of Use Requirements: The product should properly organize the data provided by the user and it should guide the user into the most optimal way for utilizing this software by providing clear guides and commands.

Personalization and Internationalization Requirements: The product should allow the user to select a chosen language. The product can be accessed from anywhere in the world.

Learning Requirements: Users should be able to quickly learn how to use the basic commands of the bot after a maximum 1 minute of reading.

Understandability and Politeness Requirements: The product shall use symbols and words that are naturally understandable by the user community. The product shall hide the details of its construction from the user (data abstraction and encapsulation).

User Documentation Requirements: The user should be able to request additional information to better understand how to utilize the software better by using the appropriate commands like !Help. The bot should also provide this documents upon joining a group chat for the first time.

### Look and Feel Requirements

The program shall be user friendly.

### Operational and Environmental Requirements

The servers of the bot should be able to run on any machine that supports interpreting python code (like Linux, Windows, Mac OS). No special hardware requirements are needed. Users should be able to interface with the product through the use of special applications like Telegram and Discord which the bot is available on. An internet connection is required.

### Cultural and Political Requirements

The product should be able to work globally.

### Legal Requirements

The source code of the product shall not be available publicly so naturally its modification by non-authorized parties is also non permitted. The confidentiality of the company’s information, data, and user’s personal information should be guaranteed.  The product shall be developed under software engineering standards.

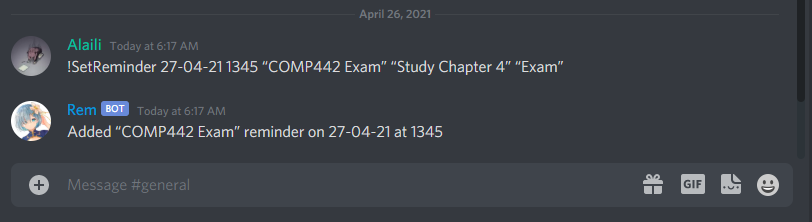
|  |  |
| --- | --- |
| Requirements ID | Description |
| **F1** | Invite the bot to group chat |
| F1.1 | Provide link to be used for the invite |
| F1.2 | Handle addition to the new group chat |
| **F2** | Handle reminder operations |
| F2.1 | Add reminder |
| F2.2 | Modify existing reminder |
| F2.3 | Delete existing reminder |
| F2.4 | View reminders |
| **F3** | Handle filtering and sorting operations |
| F3.1 | Sort/Filter reminders by date |
| F3.2 | Sort/Filter reminders by priority |
| F3.3 | Sort/Filter reminders by type |
|  |  |
| **NF** | Maximum response time of 1 second |
|  | Accurate and Precise (F2 to F3) |
|  | Ability to store large number of schedules |
|  | Should not fail |
|  | Should be available 24/7 |
|  | Data should persist |
|  | Well coded using OOP |
|  | Available on all devices supporting Telegram/Discord |
|  | Scalable |
|  | Warranty access to functions only to authorized users |
|  | Clear and simple commands |

# General Design: Bot Commands

To call the bot, we need to send a command the bot recognizes starting with an exclamation mark. Initially, we will have only 3 commands to view, set, remove reminders. (brackets mean the parameter is optional)

**!SetReminder <date> <time> <name> [<ReminderDescription> <TypeOfReminder>]**

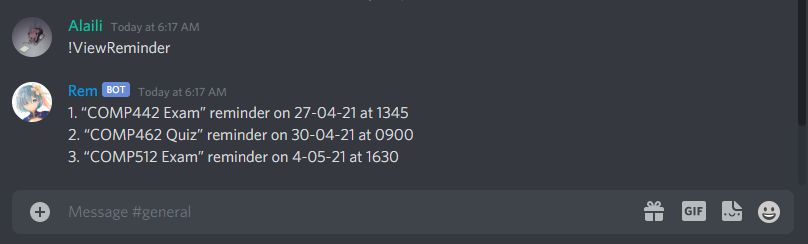
Example User Input: !SetReminder 27-04-21 1345 “COMP442 Exam” “Study Chapter 4” “Exam”  
Expected Bot Reply: Added “COMP442 Exam” reminder on 27-04-21 at 1345



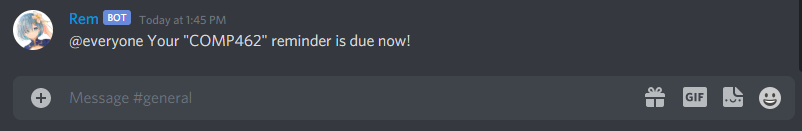
**!ViewReminder**

Example User Input: !ViewReminder  
Expected Bot Reply:

1. “COMP442 Exam” reminder on 27-04-21 at 1345  
2. “COMP462 Quiz” reminder on 30-04-21 at 0900  
3. “COMP512 Exam” reminder on 4-05-21 at 1630



**Example of Reminder being set off:**



**!RemoveReminder**

Example User Input: !RemoveReminder  
Expected Bot Reply:

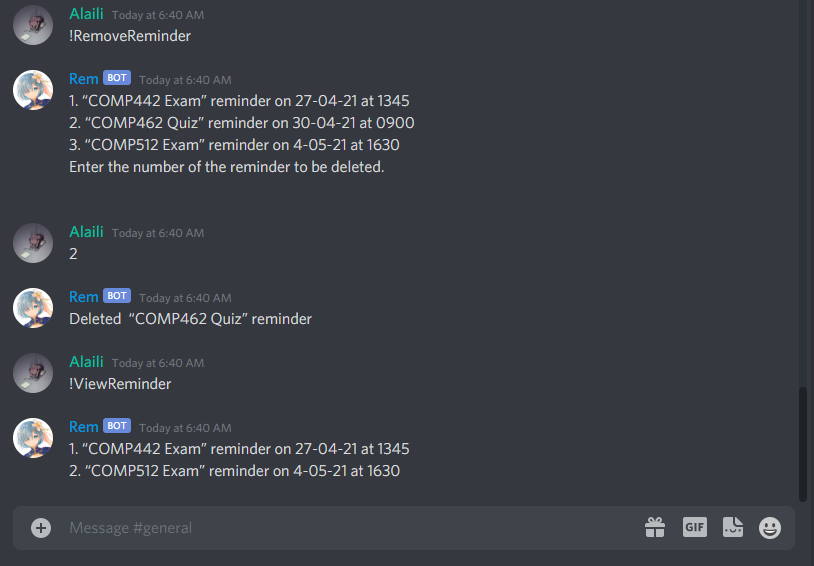
1. “COMP442 Exam” reminder on 27-04-21 at 1345  
2. “COMP462 Quiz” reminder on 30-04-21 at 0900  
3. “COMP512 Exam” reminder on 4-05-21 at 1630

Enter the number of the reminder to be deleted.

Expected User Input: 2  
Expected Bot Reply: Deleted “COMP462 Quiz” reminder

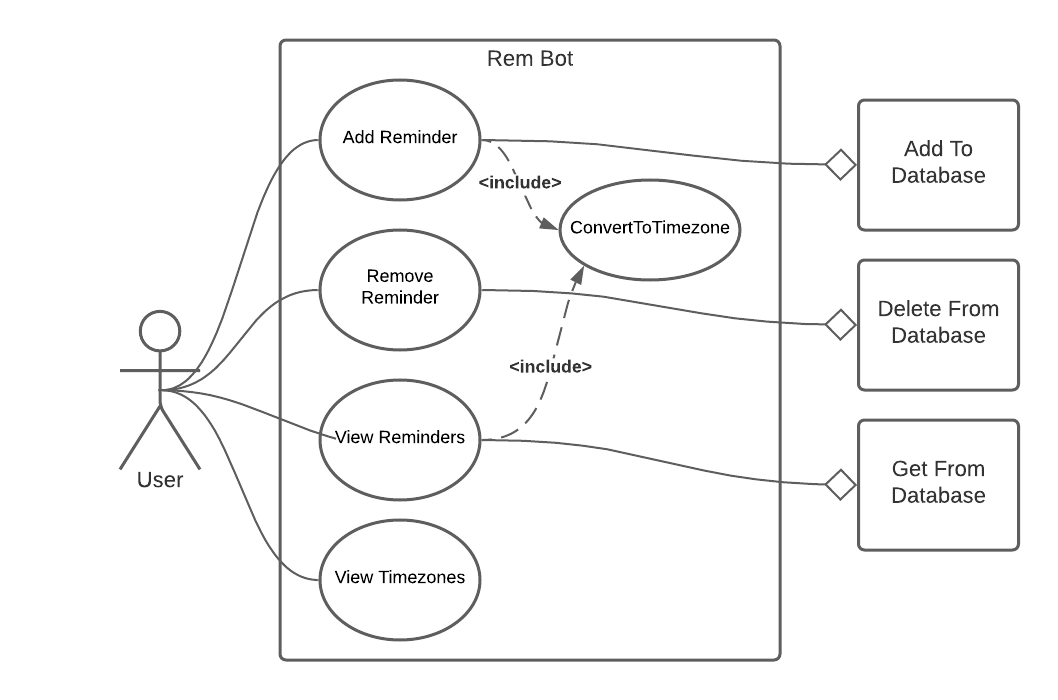
Expected User Input: !ViewReminder  
Expected Bot Reply:

1. “COMP442 Exam” reminder on 27-04-21 at 1345  
2. “COMP512 Exam” reminder on 4-05-21 at 1630



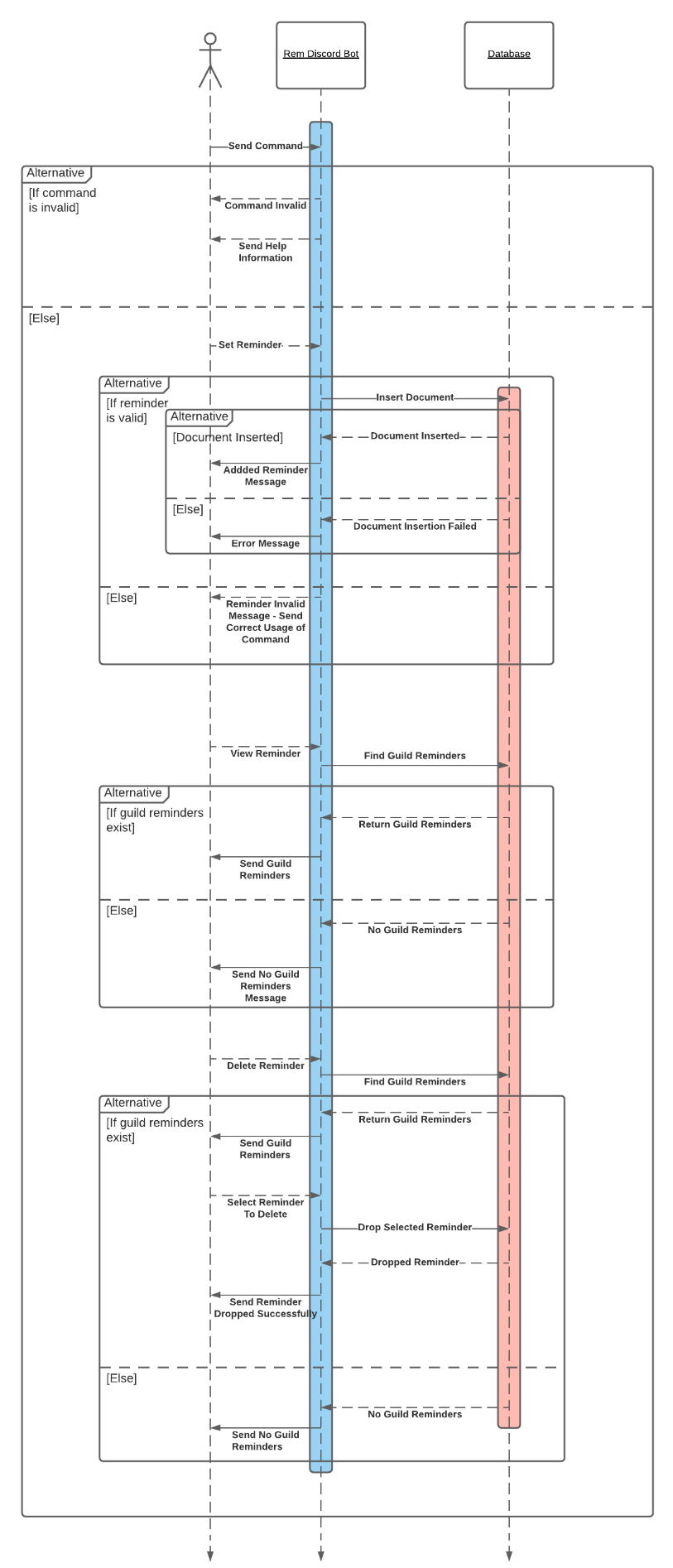
# UML Design:

## UML Use Case Design:



## UML Class Model

## UML Sequence Diagram



## UML State Machine

Rem-bot will always be active since she will always be checking for reminders and listening for commands, and there is no state to manage.

