Deliverable 1

Table of Contents

Project Title

Our project is called *RemindSpace*.

Team Members

Name	Email	GitHub Username
Ben Rust	brust203@gmail.com	B-Rust
Katie Le	katie.uyenle@gmail.com	katieuyen
Bryanth Fung	bryanthfung88@gmail.com	bry8802
Ryan Peterson	rtp4jc@gmail.com	rtp4jc
Ahmed Jallad	ahmedjallad123@gmail.com	AhmedJallad
Josh Wiedemeier	jdw170000@utdallas.edu	jdw170000
Pranjal Satija	me@pranj.co	pranjalsatija

Final Project Draft Feedback

We did not receive any explicit feedback on our initial proposal. The following announcement was posted for all project teams:

Dear All,

Thank you so much for the original ideas you proposed. Essentially, each team can go ahead and start working on their proposed project. In the meantime, please make sure that you include the following in your ***final project deliverable 2 (final deliverable)****:*

- A comprehensive research to find similar project implementations. Cite your findings properly using IEEE citation format.

- Make sure that you are adding extra feature(s) to uniquely differentiate your design from already existing similar implementations. Clearly explain what these feature(s) are.

- A comparison of your design with similar implementations in the field. This could be in any format of your choice, such as a table, paragraphs, charts, etc.

Best of luck with your projects and hope everyone enjoys working on them.

GitHub

The GitHub repository for this project can be found at https://github.com/rtp4jc/3354-RemindSpace.

Delegation of Tasks

Team Member	Assigned Tasks / Responsibilities
Ben Rust	● Help strictly define the app's capabilities. ● Help create a list of the app's functional and non-functional requirements. ● Help make the app's sequence diagrams.
Katie Le	● Collect and prepare content for the final presentation. ● Help strictly define the app's capabilities. ● Help create a list of functional and non-functional requirements for the app.
Bryanth Fung	● Help make the app's design mockups. ● Make the project scope commit to the GitHub repository. ● Help make the app's sequence diagrams.
Ryan Peterson	● Create the GitHub repository and invite other team members and the TA. ● Help strictly define the app's capabilities. ● Help create a list of the app's functional and non-functional requirements.
Ahmed Jallad	● Help make the app's design mockups. ● Help make the app's use case diagram. ● Help make the app's class / UML diagram.
Josh Wiedemeier	● Help make the app's use case diagram. ● Help make the app's sequence diagrams. ● Help make the app's class / UML diagram.
Pranjal Satija	● Select software process model and architecture. ● Manage Trello board, assignment formatting and submission, and deadlines. ● Create README for GitHub repo.

All team members will:

- Create a GitHub account.
- Join the repository.
- Contribute to project planning and direction by participating in conversations with the team.

Software Process Model

We will be using the incremental process model to develop our project. This will help us continually make small iterations so we can constantly evaluate if the project is meeting the needs we laid out for it, while making it easy to constantly adjust course as needed.

Software Requirements

Functional Requirements

1. Only the user can add, edit, and remove reminders
2. Software must send a reminder notification to the user whenever one of the following triggers are recognized by the application:
    - set time is met
    - set location is observed
    - set date is met
3. The application will always be running in the background whenever the device is on, and ready to receive signals and send notifications when reminders are triggered.
4. Reminders can be sent via phone notification, text message, or email (if the user enters a phone number and email address for the corresponding methods).
5. Users can make an account to copy and transfer all of their current reminders onto a new device.

Non-Functional Requirements

- Product Requirements
    - Usability Requirements:
        - The app should be easy to use and welcoming for new users.
        - The app should have the ability to activate more complex features and customizations for more experienced users in the settings.
    - Efficiency
        - Performance Requirements
            - Reminders should be active within 0.5 seconds of the “Set/Save” button.
            - Notifications should be sent with a latency of no greater than 5 minutes from time set or 10 minutes from the condition being satisfied for more complex triggers.
        - Space Requirements
            - The app should initially be less than 15MB. This might change in the future if machine learning has to be used in the future.
            - An individual reminder should not take up more than 0.2MB.
            - The total collection of reminders can take up as much memory as is free on the phone.
            - The reminder export should be no larger than half of the app itself at the time of the export.
    - Dependability
        - The app will only work when the phone is turned on, and will always be running in the background to check for reminder triggers.

            - The “transfer reminders to a new device” option should be available at least 98% of the time, only unavailable during server maintenance or crashes.
            - The application must be able to perform basic functions (such as sending notifications triggered by time) even when the user does not have cellular data or not connected to wifi.
        - Security
            - Reminders can be set, edited, and deleted on the app without an account.
            - The user must have an account if they wish to transfer reminders to a new device.
            - User information (usernames, passwords, and all reminder data) will be encrypted.
            - The forgotten/reset password option will result in a link being sent to the user to make a new one.
- Organizational Requirements
    - Environmental Requirements:
        - The app will be able to operate on all iOS and Android operating system versions from the past 3 years.
        - The app will be developed using Flutter to allow for cross-platform code
    - Operational Requirements:
        - The app will be used for reminders using various triggers.
    - Development Requirements:
        - The app will be written in Dart
- External Requirements
    - Regulatory Requirements:
        - The app must comply with the Google Play and Apple App Store requirements.
    - Ethical Requirements:
        - The app will not ask for unnecessary permissions.
        - The app will not mine and sell data.
        - Any data collected must have a specific internal use for improving the app and will not be exposed between users.
    - Legislative Requirements:
        - Accounting Requirements
            - The app will be available for free.
            - There will be “premium” features and there are two ways to get them: enable ads or pay a one-time fee.
        - Safety/Security Requirements
            - When available in Europe, the app must be compliant with the European Union’s General Data Protection Regulation (GDPR).

Architecture

We will be using the MVC architecture for our project. We chose this architecture because it adequately provides a method that is able to handle the scale and complexity of the project as it grows and evolves over time, including the usage of multiple isolated triggers. MVC supports rapid and efficient implementations during the development of the application, allowing multiple programmers to concurrently work on an aspect of the model; one programmer can build the views while another work on the controller. Furthermore, since the model itself does not depend on only the views aspect, we can alter the appearance of the application without changing its entire architecture. MVC is also the architecture recommended by Apple and Google for building apps on their platforms. In addition, MVC is the only architecture that explicitly mentions a user interface, which is a key part of our project.

The repository model could apply to our project, and elements of it will when it comes to manipulating and accessing data, but the architecture as a whole does not fit the design of a mobile app.

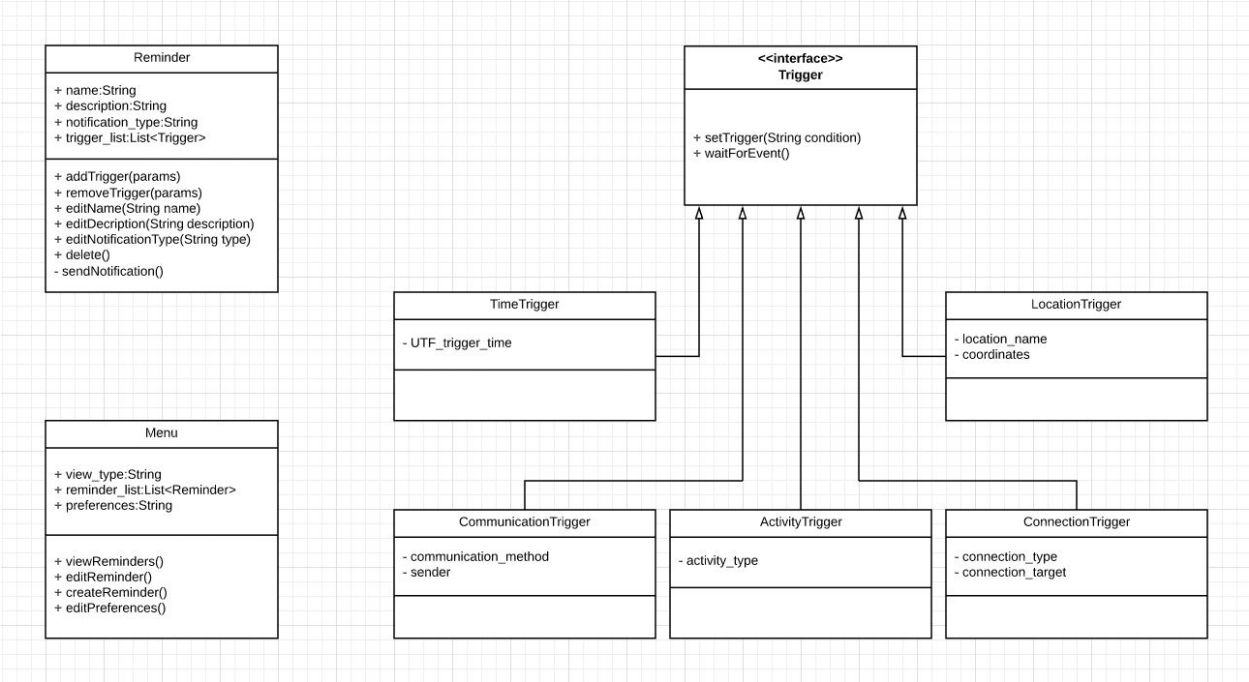
The layered architecture doesn't make a lot of sense for us, as the model for our app doesn't cleanly divide into vertical layers that build on top of one another.

Our project will incorporate elements of the client-server model, as the app will be storing copies of user data in the server so it's accessible across multiple devices, but the app itself will not be built using that architecture.

The pipe and filter model doesn't work well for our project because we aren't doing any extensive data processing.

Class Diagram

The class diagram is available on LucidChart at the following link: https://www.lucidchart.com/invitations/accept/e656be60-ade9-48eb-bee8-94c84ef9e231. It is also attached in image form below.



Use Case Diagrams

The use case diagrams are available on LucidChart at the following link:
https://www.lucidchart.com/invitations/accept/c06e4f20-0d88-40e8-9cc2-684957a25eba

List of Use Cases

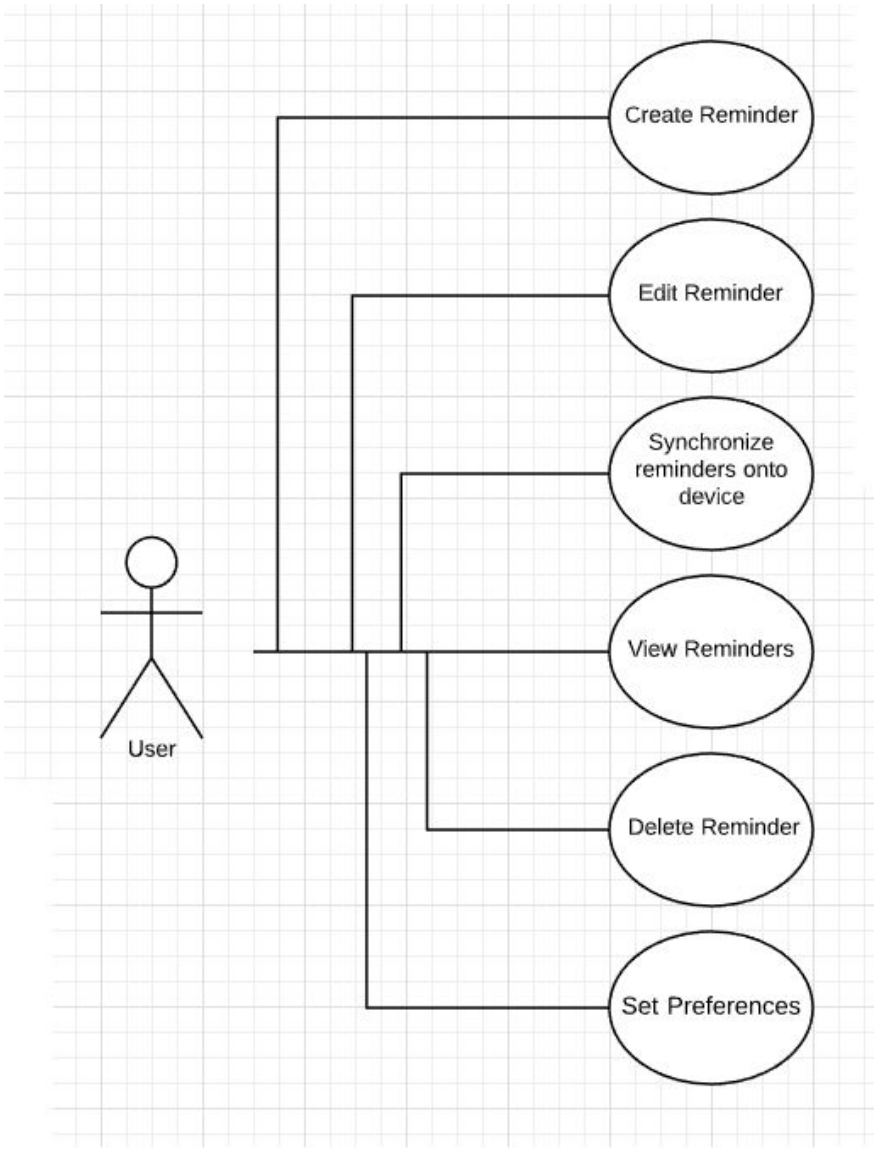
- Actors/Stick figures in diagrams:
    a. Person
    b. RemindSpace app
    c. Phone system

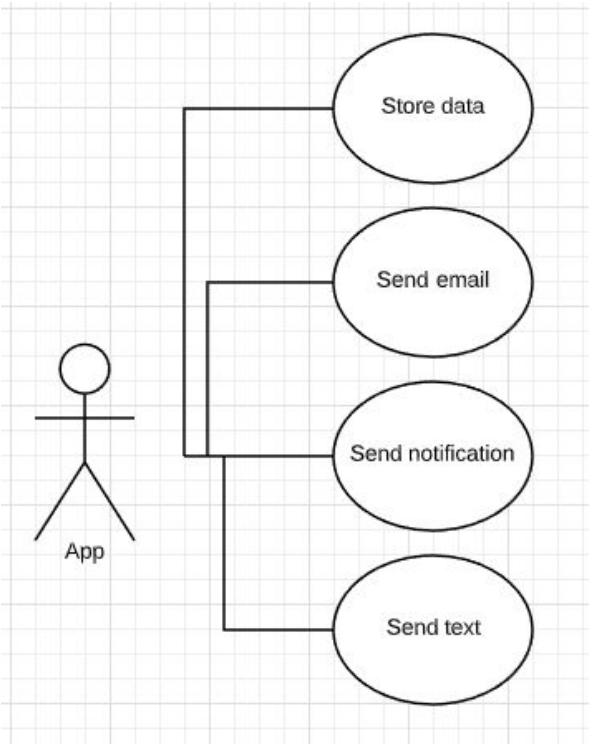
- Sequences for a Person
    1. Make a reminder
    2. Edit reminder
    3. Synchronize between multiple devices
    4. View existing reminders
    5. Delete reminder
    6. Set preferences of the app (change colors, background, texts, etc.)

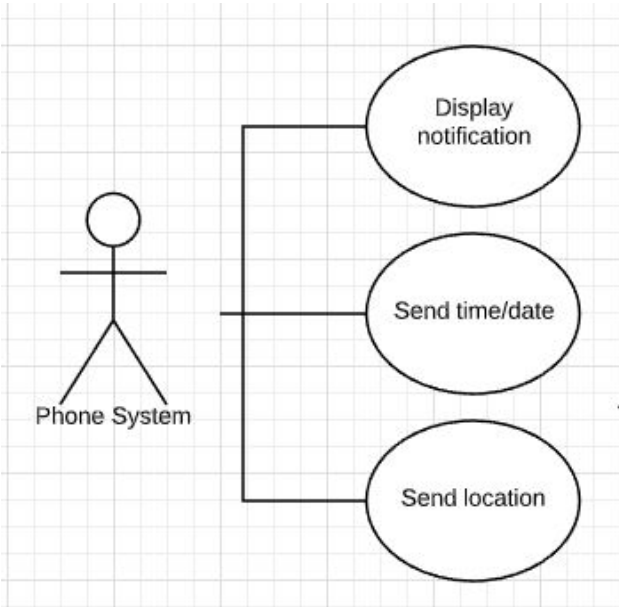
- Sequences for the App:
    1. Send notification to phone - Done to phone
    2. Send SMS to User
    3. Send email to User
    4. Store data

- Sequences for the Phone System:
    1. Send time / date
    2. Send location
    3. Display notification

Image Forms of Diagrams




Store data
Send email
Send notification
Send text
App


Display notification
Send time/date
Send location
Phone System

Sequence Diagrams

The sequence diagrams are available on LucidChart at this link: https://www.lucidchart.com/invitations/accept/e8745314-64cc-4509-93fa-67f31d2b17f4. They are also attached in image form below.

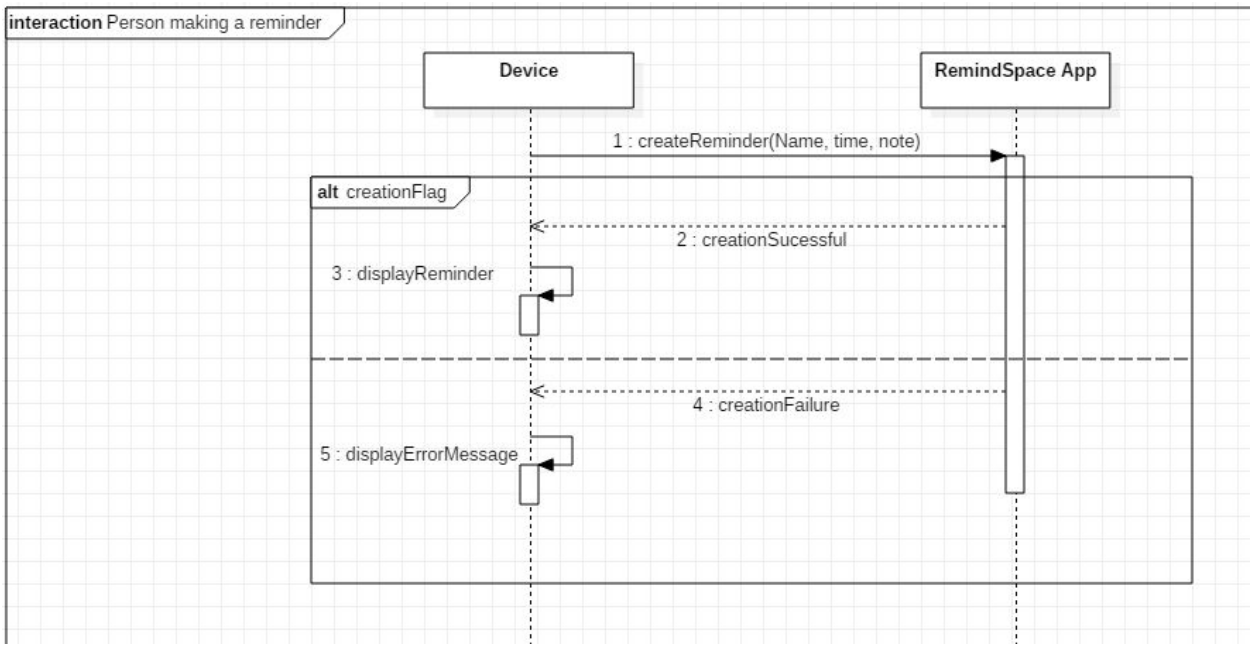
List of Sequences

1. Person making a reminder
2. Person editing a reminder
3. Person synchronizing reminders onto other devices
4. Person viewing existing reminders
5. Person deleting reminder
6. Person setting preferences in the app

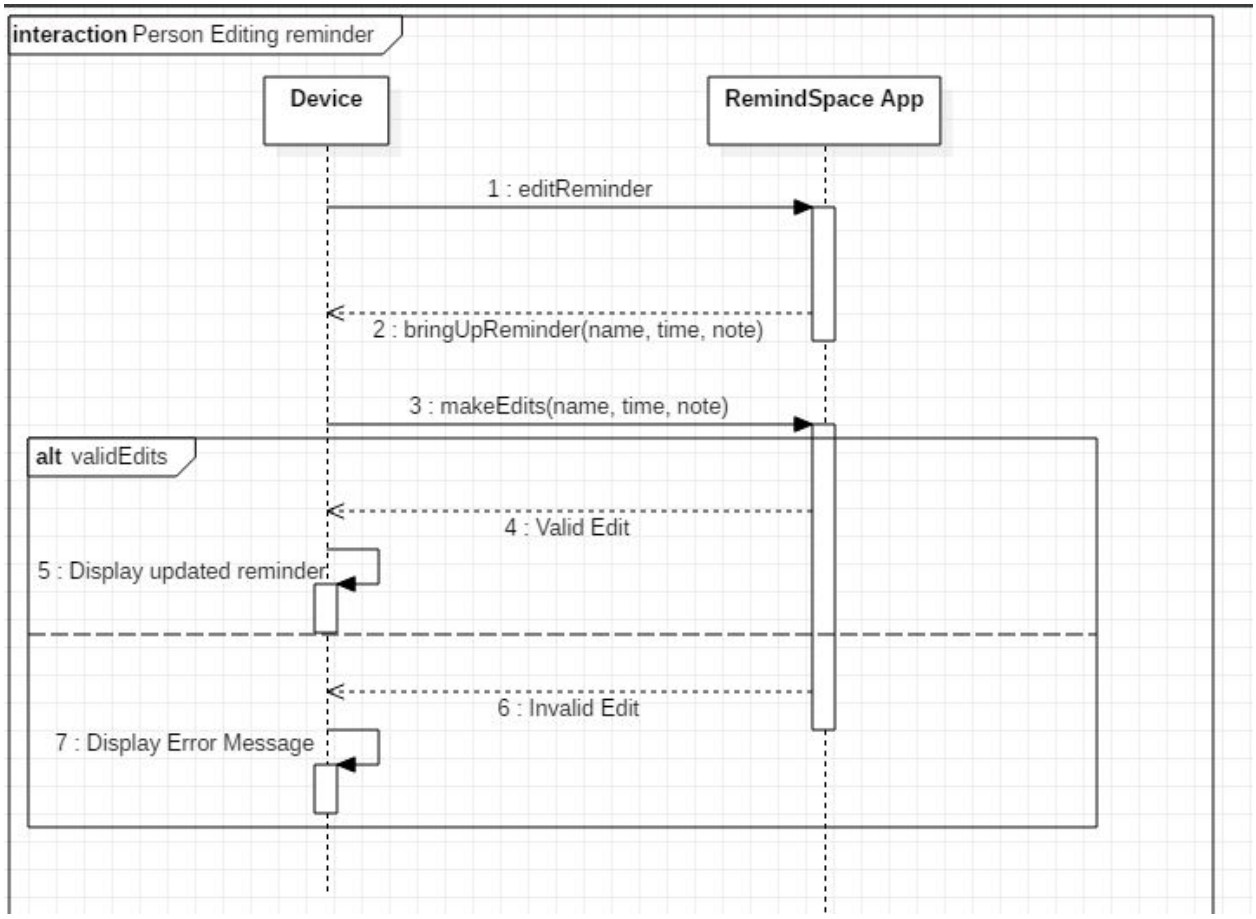
7. App sending notifications to phone
8. App sending SMS to user
9. App sending email to user
10. App storing data

11. Phone system sending time/date to app
12. Phone system sending location to app
13. Display notification

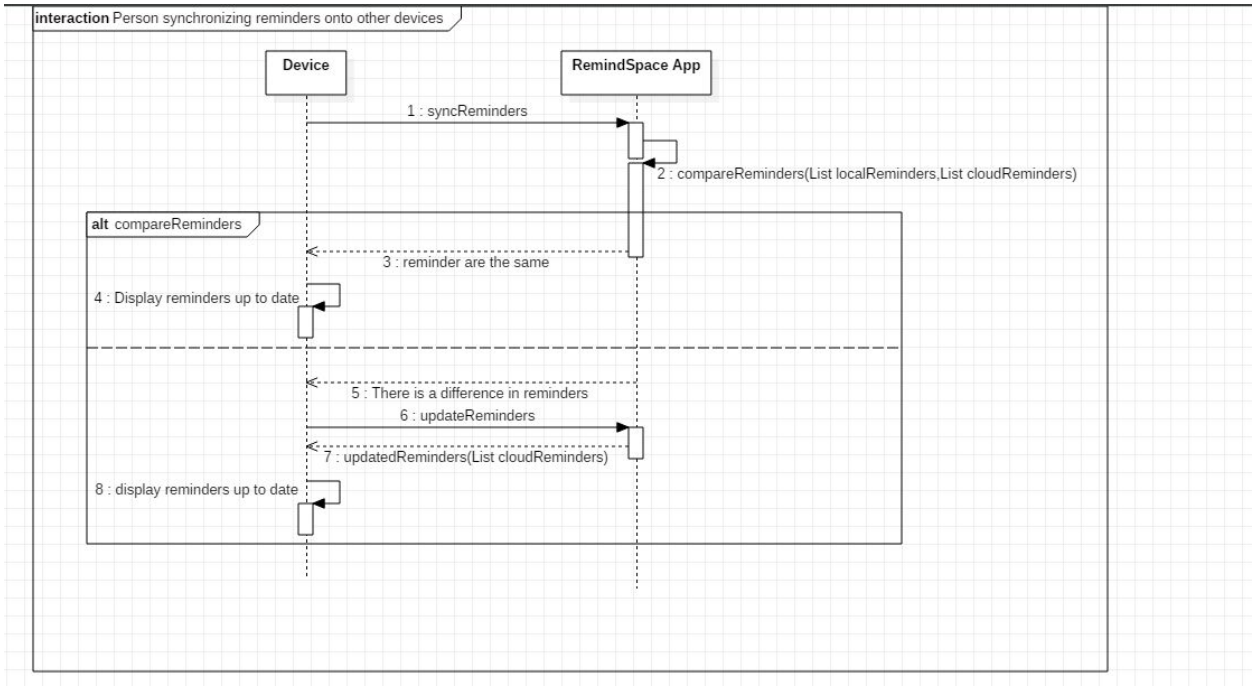
Images

1. 
interaction Person making a reminder
Device
RemindSpace App
1 : createReminder(Name, time, note)
alt creationFlag
2 : creationSucessful
3 : displayReminder
4 : creationFailure
5 : displayErrorMessage

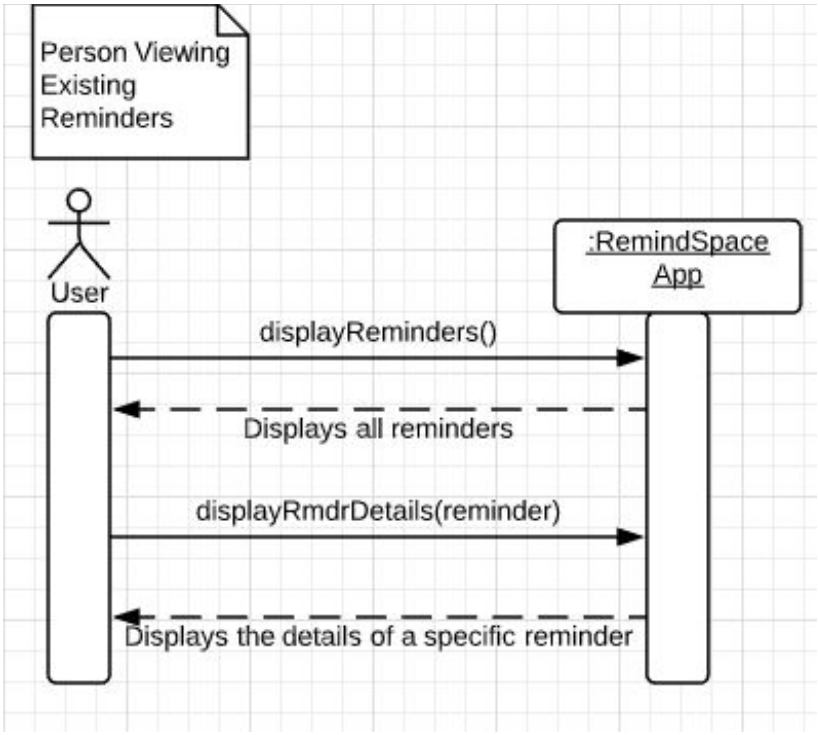
2.


interaction Person Editing reminder
Device
RemindSpace App
1 : editReminder
2 : bringUpReminder(name, time, note)
3 : makeEdits(name, time, note)
alt validEdits
4 : Valid Edit
5 : Display updated reminder
6 : Invalid Edit
7 : Display Error Message

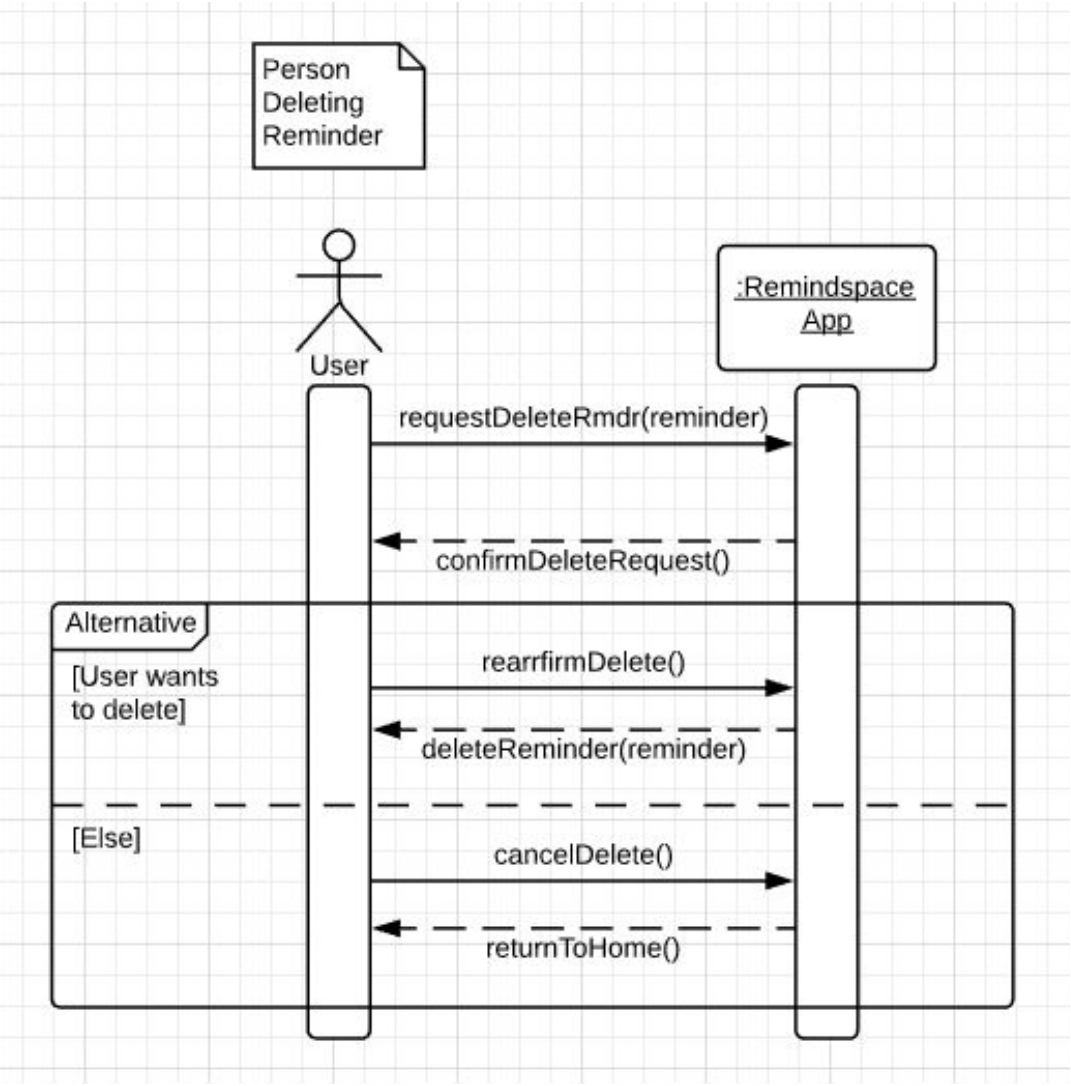
3.


interaction Person synchronizing reminders onto other devices
Device
RemindSpace App
1 : syncReminders
2 : compareReminders(List localReminders,List cloudReminders)
alt compareReminders
3 : reminder are the same
4 : Display reminders up to date
5 : There is a difference in reminders
6 : updateReminders
7 : updatedReminders(List cloudReminders)
8 : display reminders up to date

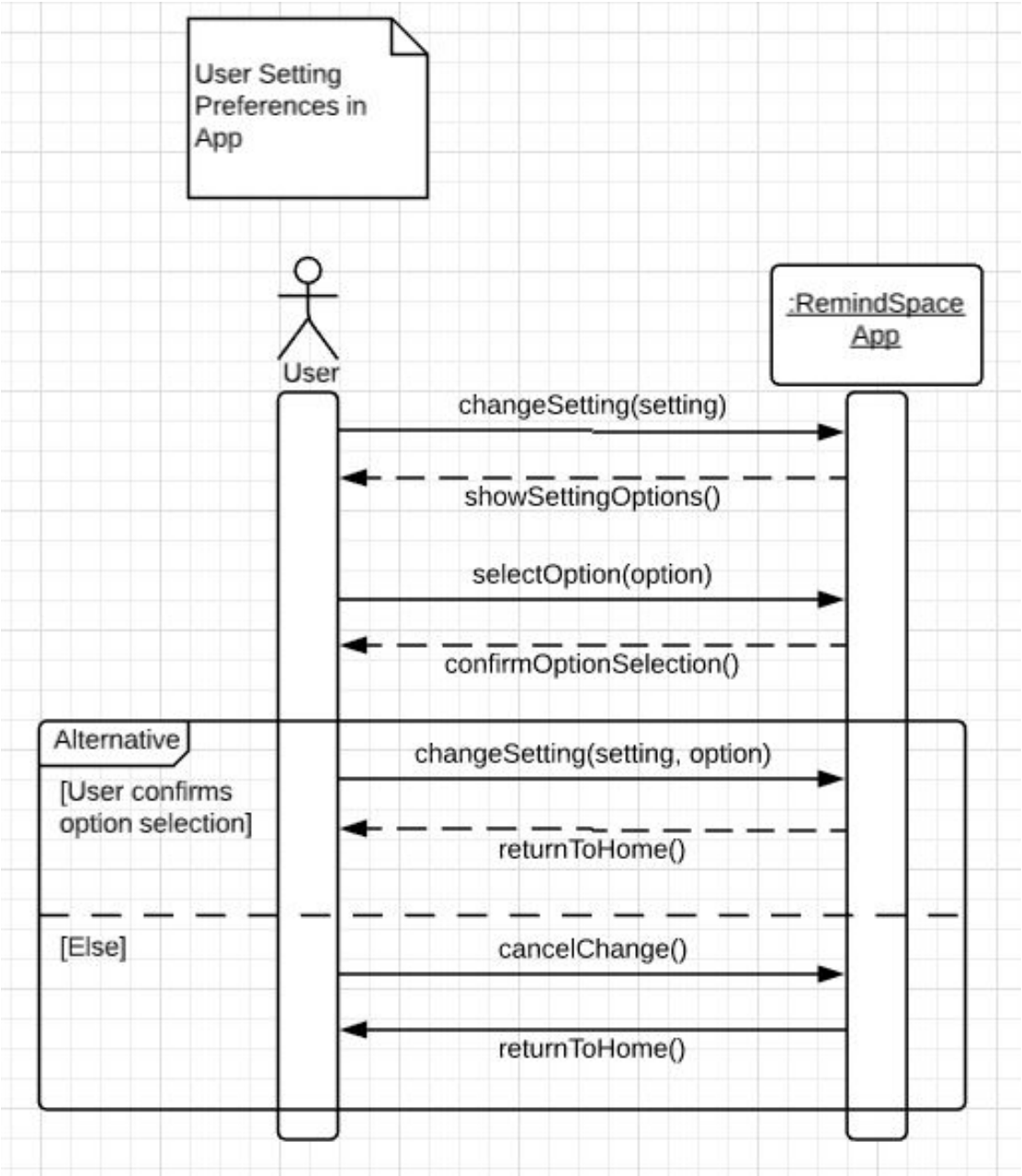
4.


Person Viewing Existing Reminders
User
:RemindSpace App
displayReminders()
Displays all reminders
displayRmdrDetails(reminder)
Displays the details of a specific reminder

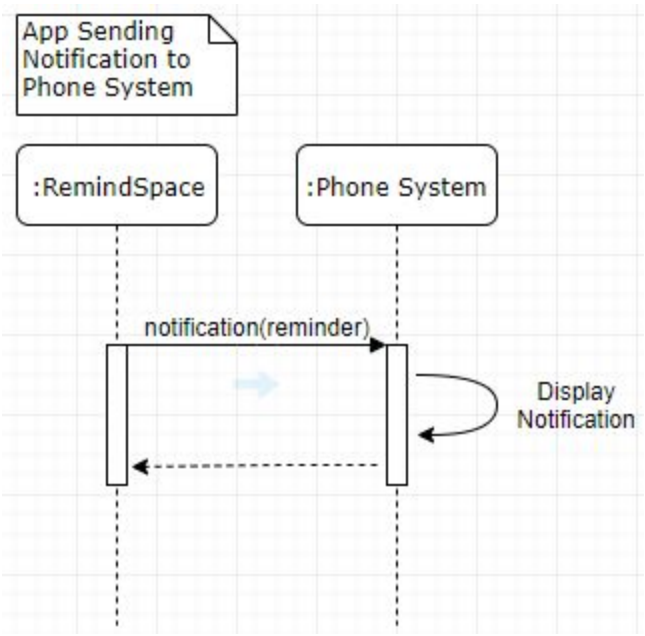
5.


Person
Deleting
Reminder
User
:Remindspace
App
requestDeleteRmdr(reminder)
confirmDeleteRequest()
Alternative
[User wants
to delete]
rearrfirmDelete()
deleteReminder(reminder)
[Else]
cancelDelete()
returnToHome()

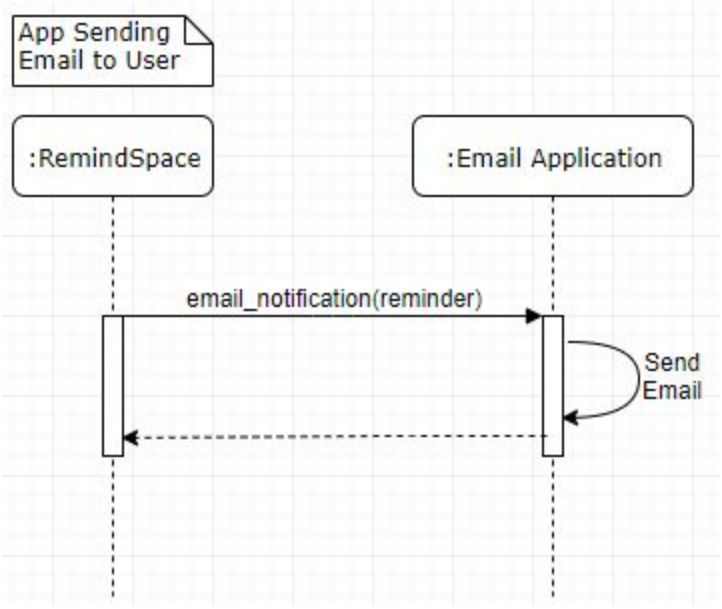
6.


User Setting Preferences in App
User
:RemindSpace App
changeSetting(setting)
showSettingOptions()
selectOption(option)
confirmOptionSelection()
Alternative
[User confirms option selection]
changeSetting(setting, option)
returnToHome()
[Else]
cancelChange()
returnToHome()

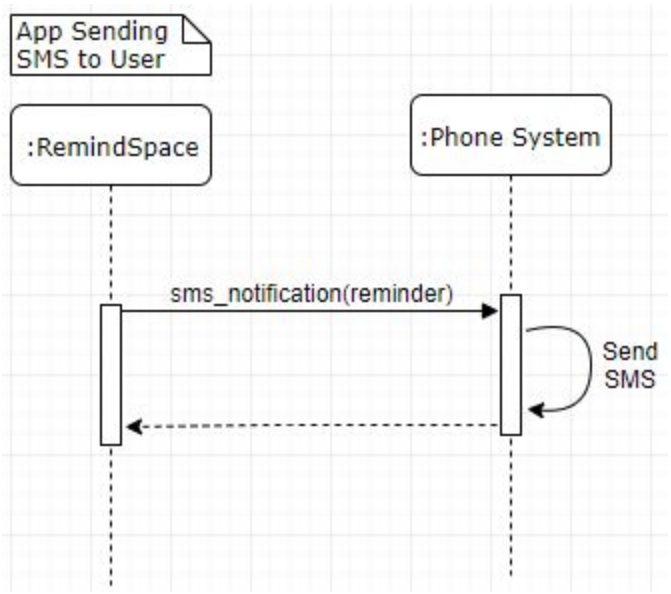
7.


App Sending
Notification to
Phone System
:RemindSpace
:Phone System
notification(reminder)
Display
Notification

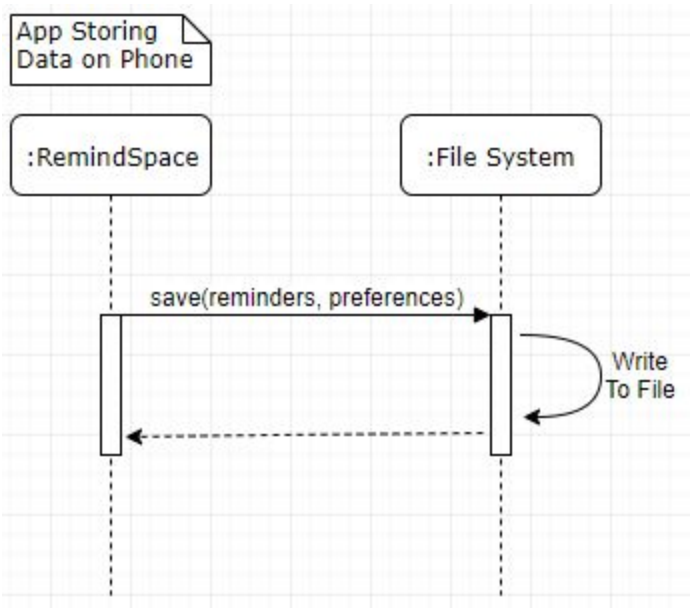
8.


App Sending
Email to User
:RemindSpace
:Email Application
email_notification(reminder)
Send
Email

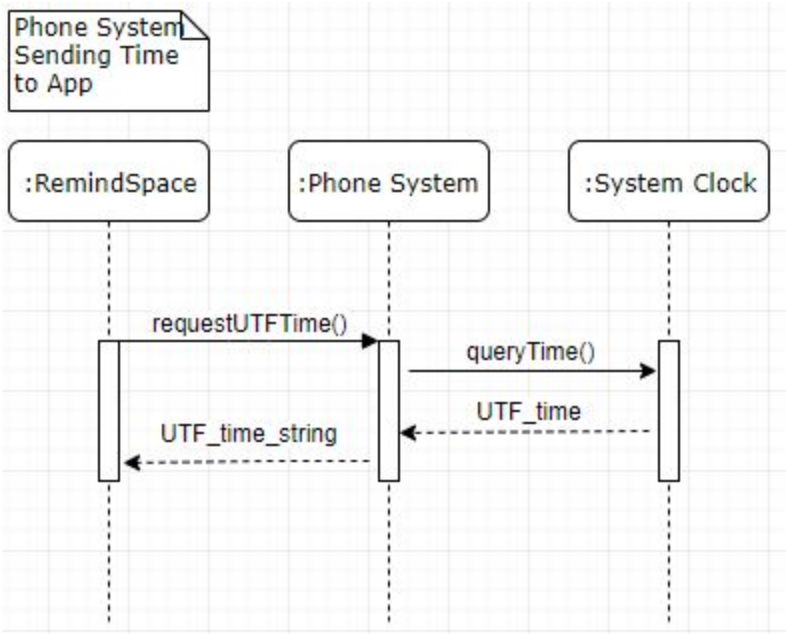
9.


App Sending
SMS to User
:RemindSpace
:Phone System
sms_notification(reminder)
Send
SMS

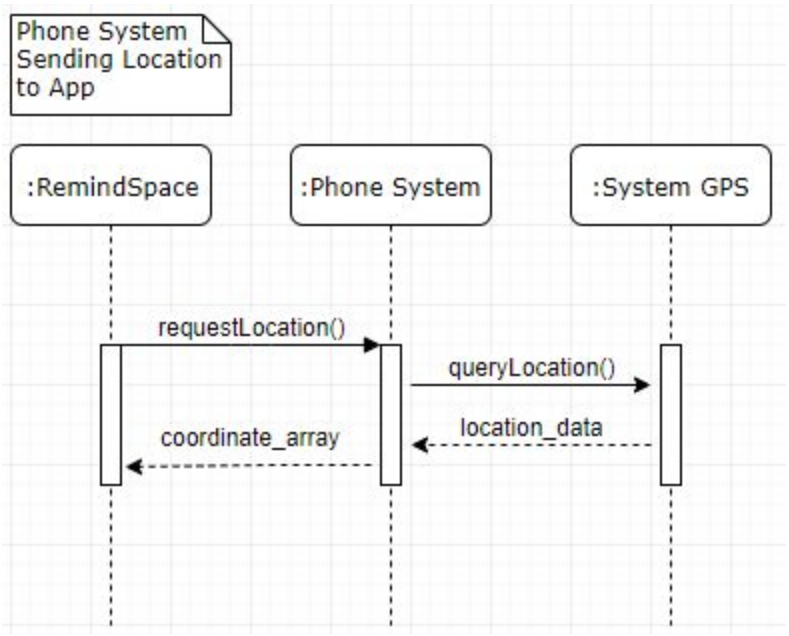
10.


App Storing
Data on Phone
:RemindSpace
:File System
save(reminders, preferences)
Write
To File

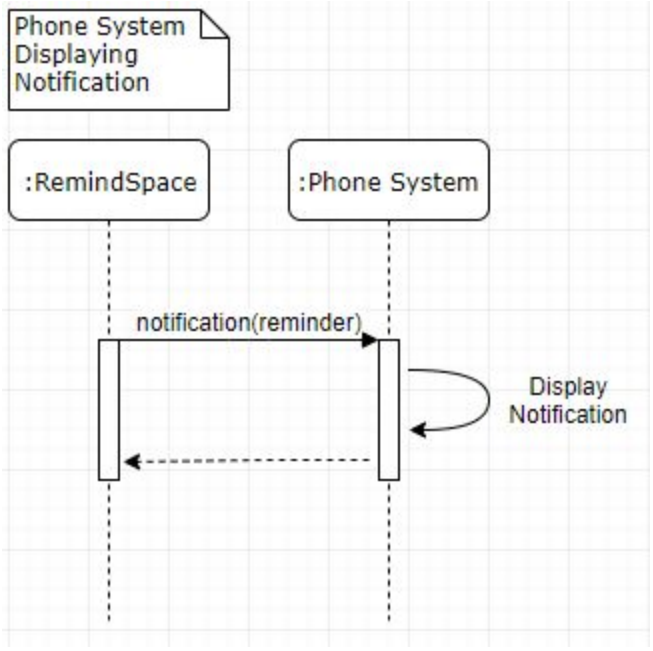
11.


Phone System
Sending Time
to App
:RemindSpace
:Phone System
:System Clock
requestUTFTime()
queryTime()
UTF_time
UTF_time_string

12.


Phone System
Sending Location
to App
:RemindSpace
:Phone System
:System GPS
requestLocation()
queryLocation()
location_data
coordinate_array

13.


Phone System
Displaying
Notification
:RemindSpace
:Phone System
notification(reminder)
Display
Notification