Scribbles

By : Symbat, Saamia, Sanjana, Levith

Team Introduction

Symbat Bezhigit

Software Engineer

Saamia Shafqat

Software Engineer

Sanjana Nambiar

Software Engineer

Levith Andrade Cuellar

Software Engineer

Project Objectives

What our project aims to achieve.

1. **Combat "silo mentality"** by creating a system of communication that focuses on integration and centralization.

2. **Bolster successful medical handoffs** by creating a system that can streamline communication as recommended by the HIPAA Journal.

3. **Make information easily accessible to healthcare professionals** via an intuitive and readily accessible interface.

Scope Definition

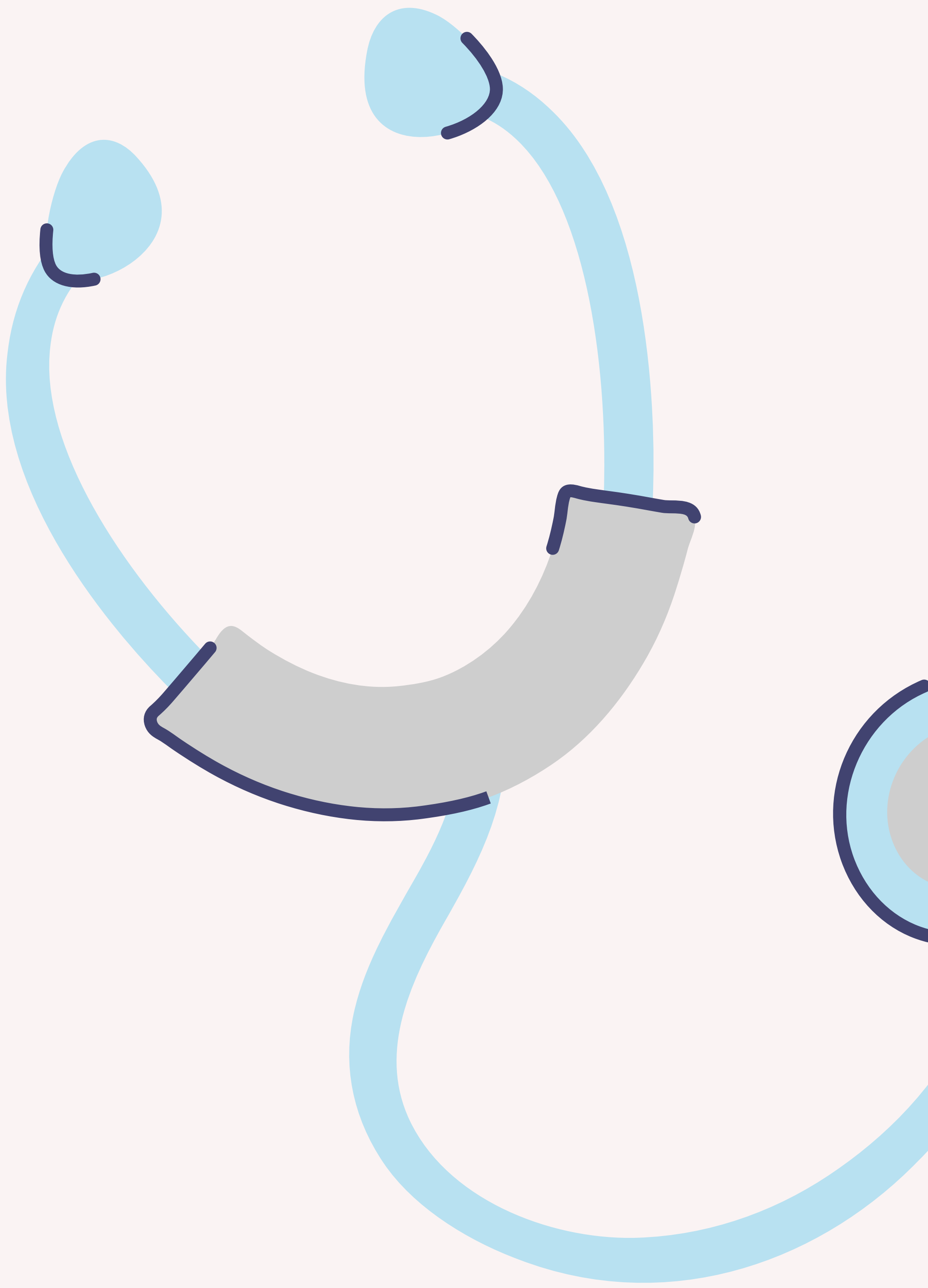
Boundaries and limitations of our project

Inside

Messaging, Pager, Patient Record Database and Remote Access, AI-Supported Analysis and Reminder Setting.

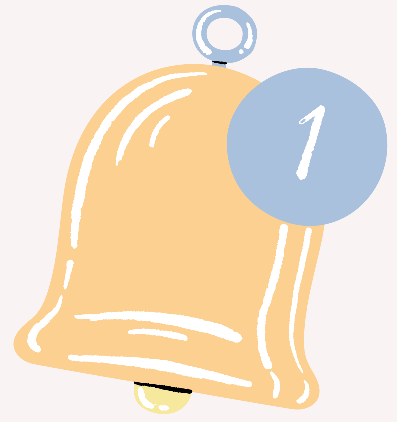
Outside

Health Diagnosis, Administrative Work, Communication with Patients, Complete Access to Patient Information, Patient Appointments.



User Requirements

It has to be ***mobile****! I don't want to have to sit at my desk to access these features!*



I want to have ***everything in one place.*** *I get so many notifications!*

Consulting with other doctors *is an important part of my job!*

Having remote access to patient records is my ***number one feature****.*

Dr. Iliana Cuellar

Ear. Nose and Throat Doctor

Love the pager, but ***differentiate*** *from the chat feature.*

Functional Requirements

Messaging

To communicate short messages regarding patient care with each other.

Paging

To page or call each other to a certain location inside the hospital.

EHR Viewing

To view patient records while caring for a patient.

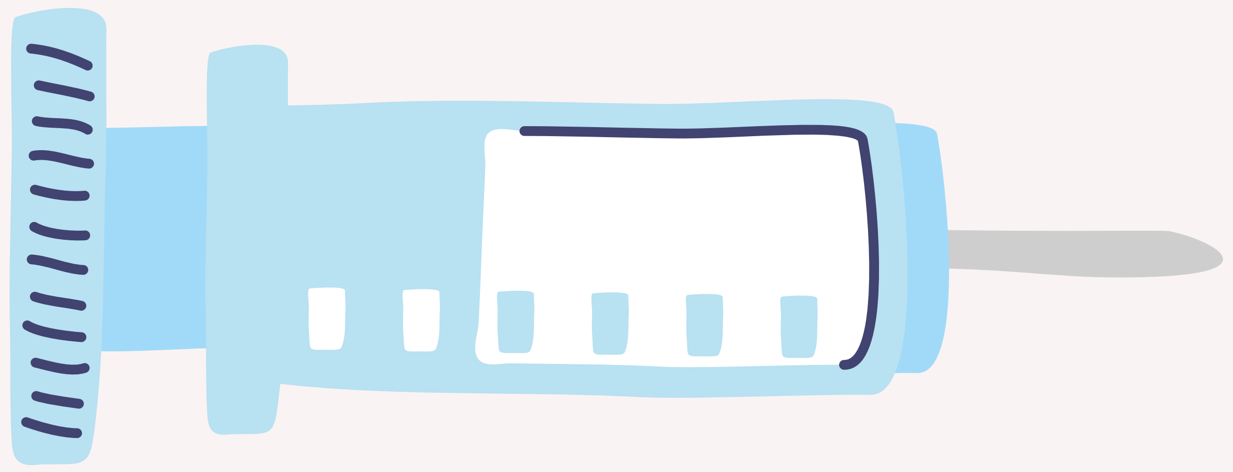
Reminders

To manually create reminders of scheduled visits, times to give medications to patients, etc.

AI-Integration

To make the process of reviewing patient records more efficient.

Non-Functional Requirements



Security

The system shall enforce **data encryption in transit and at rest when querying patient information** using industry-standard cryptographic protocols (e.g., TLS for data in transit, AES-256 for data at rest), with regular encryption key rotation conducted every 90 days to enhance data security.

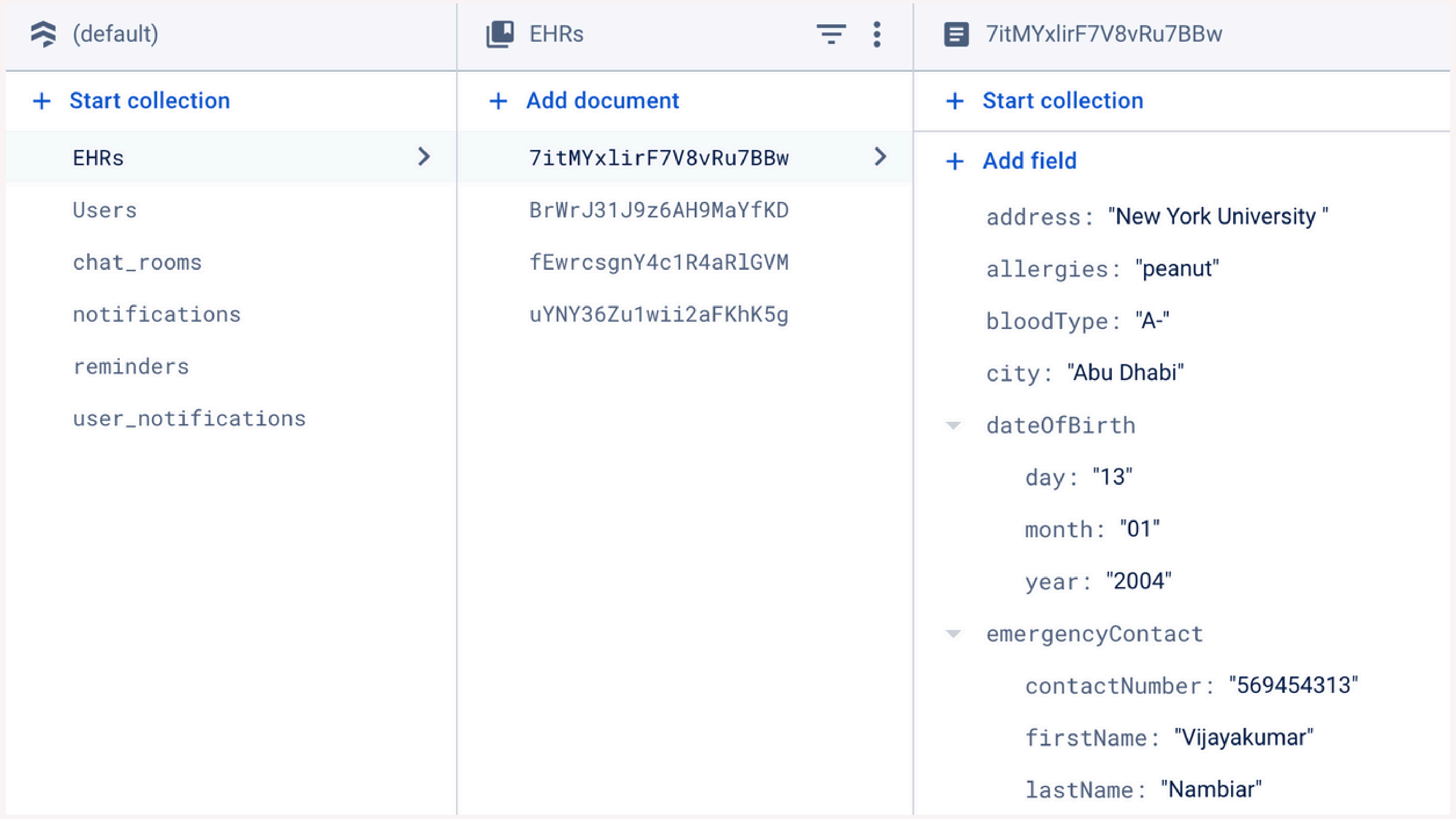
Privacy

The system shall only provide **anonymized patient information to the AI summary feature** to comply with relevant healthcare data protection regulations such as HIPAA.

Performance

The system shall **respond to a query of the patient database immediately** This response time is considered a maximum limit under normal operating conditions.

System Architecture

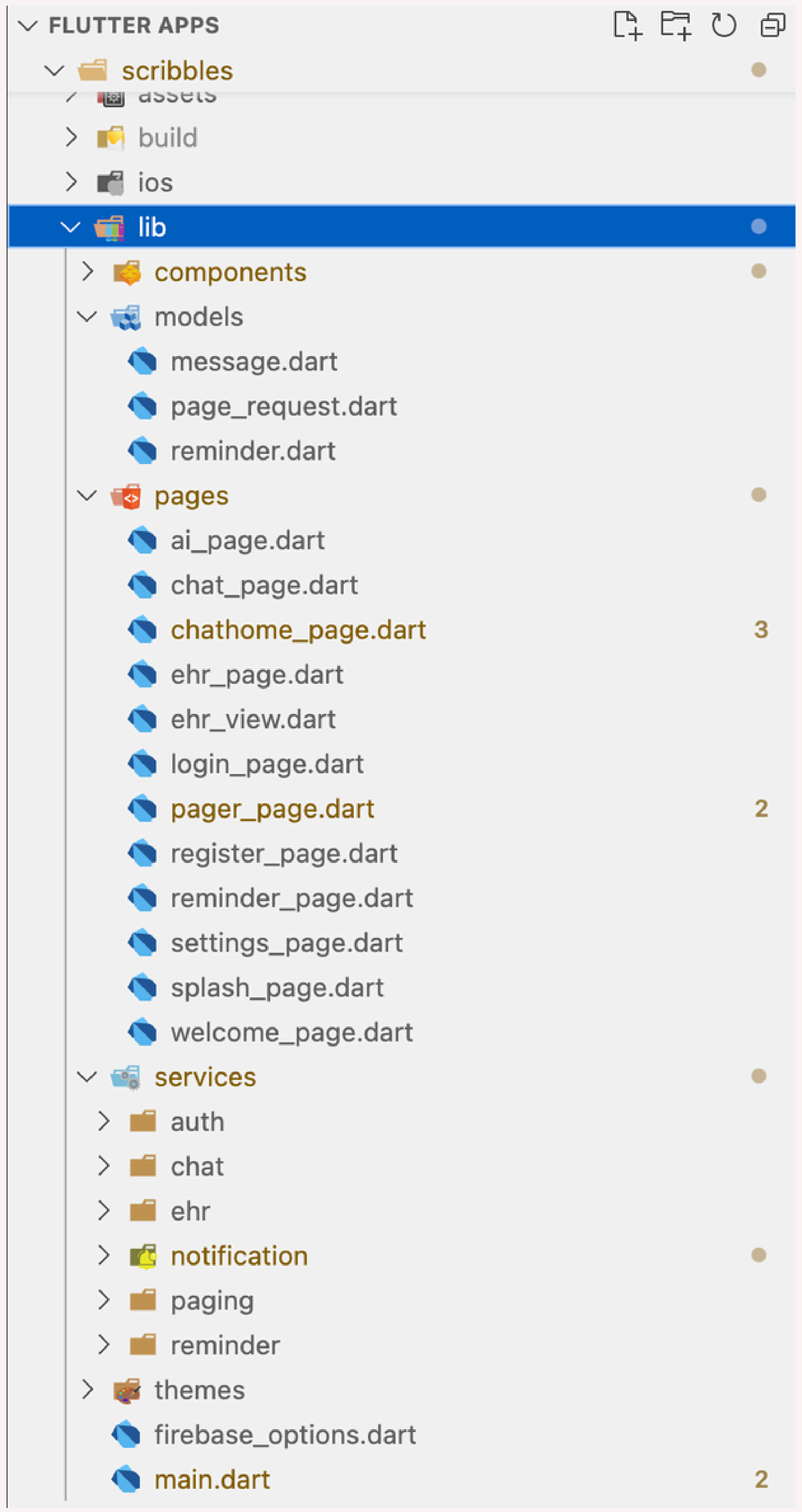


(default)
+ Start collection
EHRs
Users
chat_rooms
notifications
reminders
user_notifications

EHRs
+ Add document
7itMYxlirF7V8vRu7BBw
BrWrJ31J9z6AH9MaYfKD
fEwrcsgnY4c1R4aRlGVM
uYNY36Zu1wii2aFKhK5g

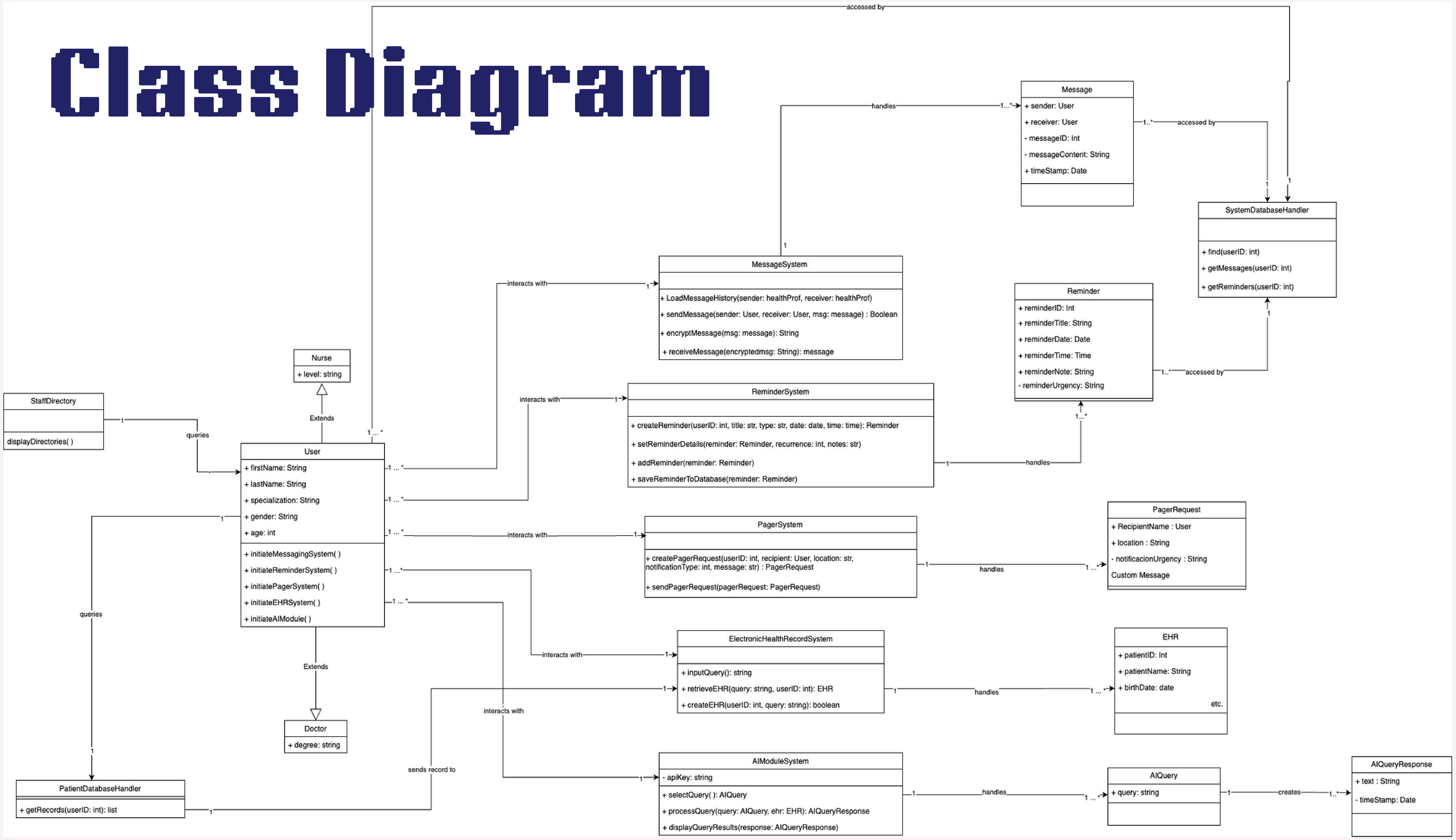
7itMYxlirF7V8vRu7BBw
+ Start collection
+ Add field

```
address: "New York University "
allergies: "peanut"
bloodType: "A-"
city: "Abu Dhabi"
dateOfBirth
    day: "13"
    month: "01"
    year: "2004"
emergencyContact
    contactNumber: "569454313"
    firstName: "Vijayakumar"
    lastName: "Nambiar"
```

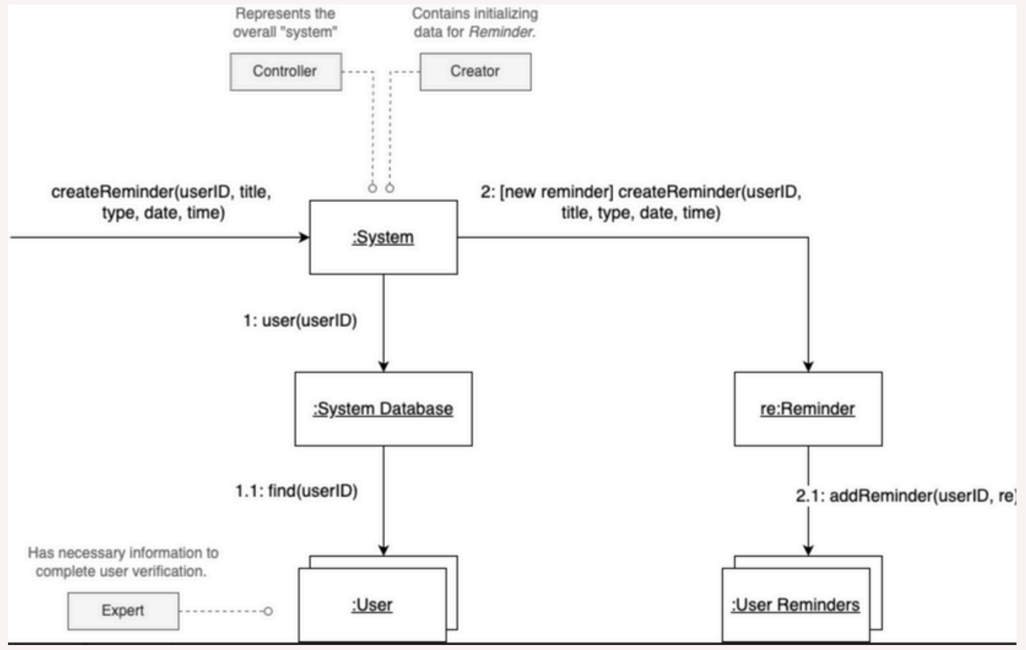


```
FLUTTER APPS
  scribbles
    assets
    build
    ios
    lib
      components
      models
        message.dart
        page_request.dart
        reminder.dart
      pages
        ai_page.dart
        chat_page.dart
        chathome_page.dart        3
        ehr_page.dart
        ehr_view.dart
        login_page.dart
        pager_page.dart           2
        register_page.dart
        reminder_page.dart
        settings_page.dart
        splash_page.dart
        welcome_page.dart
      services
        auth
        chat
        ehr
        notification
        paging
        reminder
      themes
      firebase_options.dart
      main.dart                   2
```

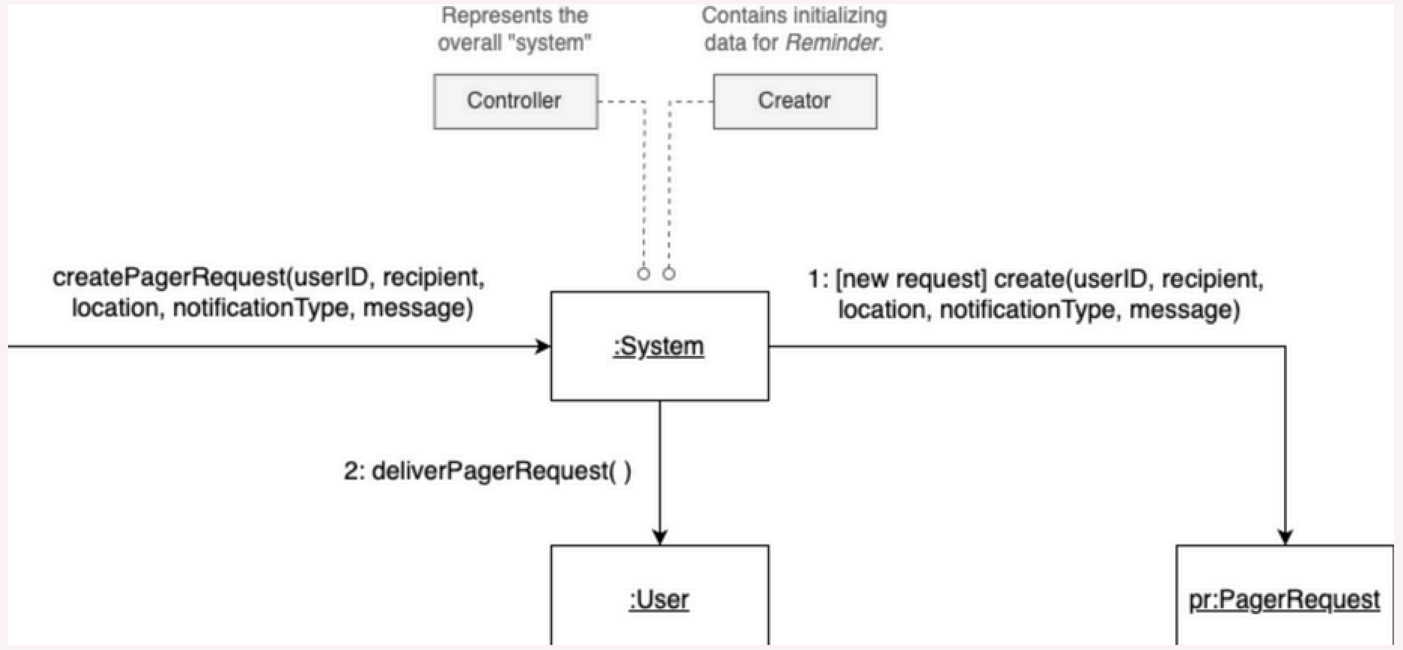
Class Diagram


accessed by
Message
+ sender: User
+ receiver: User
- messageID: Int
- messageContent: String
+ timeStamp: Date
handles
1...*
1..*
accessed by
1
1
SystemDatabaseHandler
+ find(userID: int)
+ getMessages(userID: int)
+ getReminders(userID: int)
1
MessageSystem
+ LoadMessageHistory(sender: healthProf, receiver: healthProf)
+ sendMessage(sender: User, receiver: User, msg: message) : Boolean
+ encryptMessage(msg: message): String
+ receiveMessage(encryptedmsg: String): message
interacts with
Reminder
+ reminderID: Int
+ reminderTitle: String
+ reminderDate: Date
+ reminderTime: Time
+ reminderNote: String
- reminderUrgency: String
1..*
accessed by
1...*
Nurse
+ level: string
Extends
StaffDirectory
displayDirectories( )
queries
1 ... *
User
+ firstName: String
+ lastName: String
+ specialization: String
+ gender: String
+ age: int
+ initiateMessagingSystem( )
+ initiateReminderSystem( )
+ initiatePagerSystem( )
+ initiateEHRSystem( )
+ initiateAIModule( )
interacts with
ReminderSystem
+ createReminder(userID: int, title: str, type: str, date: date, time: time): Reminder
+ setReminderDetails(reminder: Reminder, recurrence: int, notes: str)
+ addReminder(reminder: Reminder)
+ saveReminderToDatabase(reminder: Reminder)
handles
PagerRequest
+ RecipientName : User
+ location : String
- notificacionUrgency : String
Custom Message
interacts with
PagerSystem
+ createPagerRequest(userID: int, recipient: User, location: str, notificationType: int, message: str) : PagerRequest
+ sendPagerRequest(pagerRequest: PagerRequest)
handles
queries
interacts with
ElectronicHealthRecordSystem
+ inputQuery(): string
+ retrieveEHR(query: string, userID: int): EHR
+ createEHR(userID: int, query: string): boolean
handles
EHR
+ patientID: Int
+ patientName: String
+ birthDate: date
etc.
Extends
Doctor
+ degree: string
interacts with
sends record to
AIModuleSystem
- apiKey: string
+ selectQuery( ): AIQuery
+ processQuery(query: AIQuery, ehr: EHR): AIQueryResponse
+ displayQueryResults(response: AIQueryResponse)
handles
AIQuery
+ query: string
creates
AIQueryResponse
+ text : String
- timeStamp: Date
PatientDatabaseHandler
+ getRecords(userID: int): list

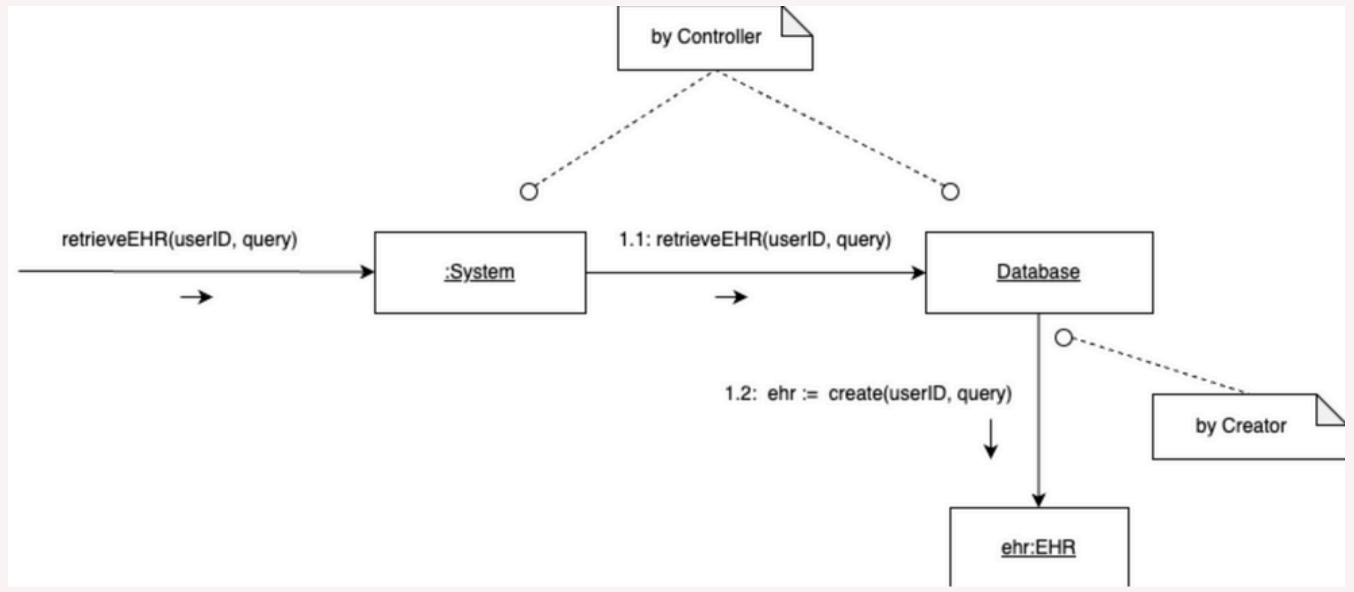
Interaction Diagrams


Represents the overall "system"
Contains initializing data for *Reminder*.
Controller
Creator
createReminder(userID, title, type, date, time)
2: [new reminder] createReminder(userID, title, type, date, time)
:System
1: user(userID)
:System Database
re:Reminder
1.1: find(userID)
2.1: addReminder(userID, re
Has necessary information to complete user verification.
Expert
:User
:User Reminders

Reminder

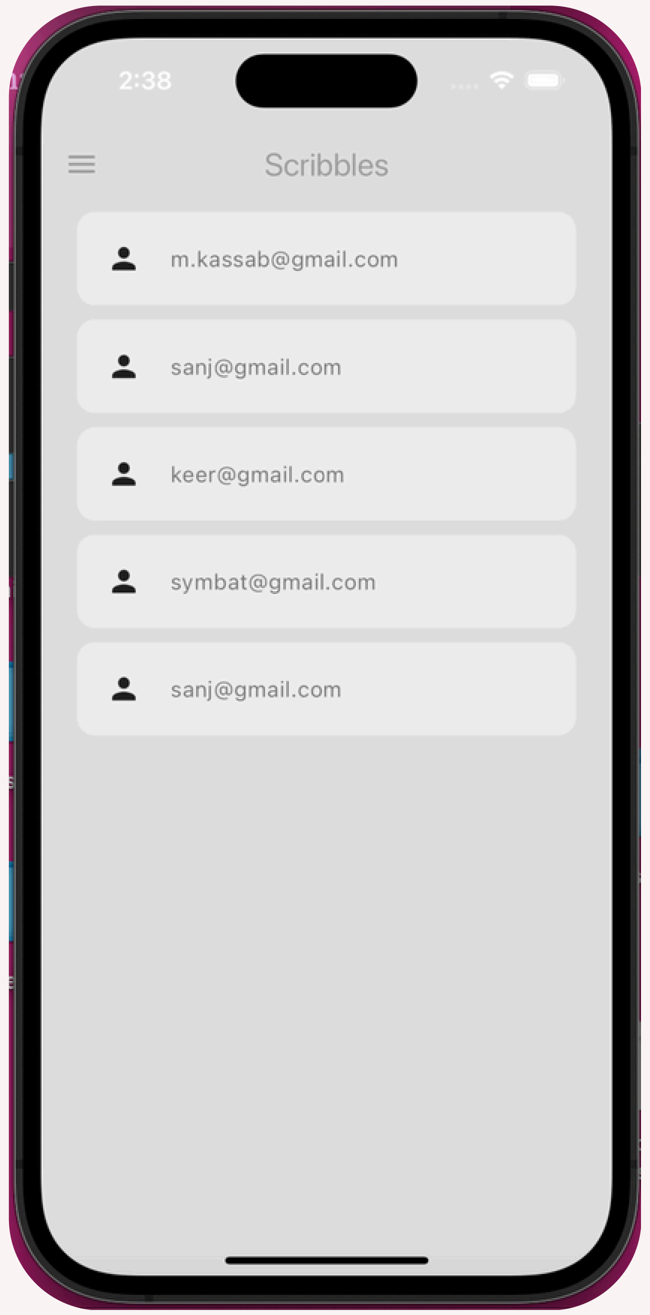

Represents the overall "system"
Contains initializing data for *Reminder*.
Controller
Creator
createPagerRequest(userID, recipient, location, notificationType, message)
1: [new request] create(userID, recipient, location, notificationType, message)
:System
2: deliverPagerRequest( )
:User
pr:PagerRequest

Pager


by Controller
retrieveEHR(userID, query)
:System
1.1: retrieveEHR(userID, query)
Database
1.2: ehr := create(userID, query)
by Creator
ehr:EHR

View EHR

Messaging



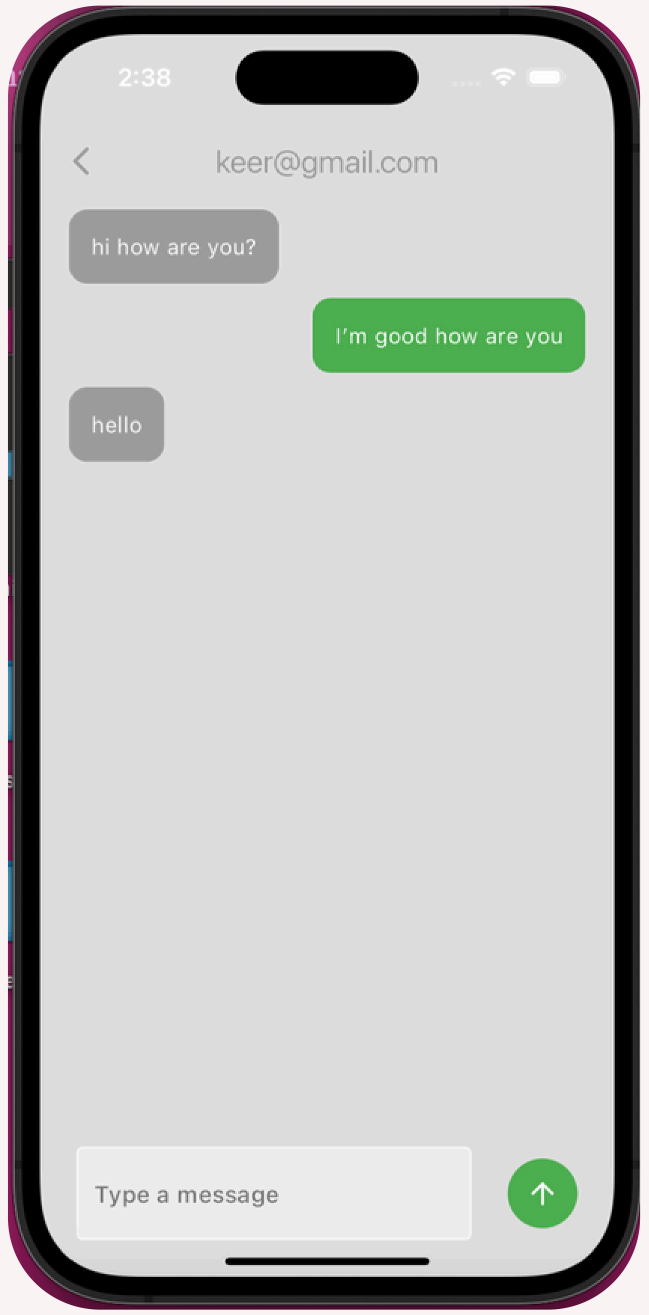
m.kassab@gmail.com

sanj@gmail.com

keer@gmail.com

symbat@gmail.com

sanj@gmail.com

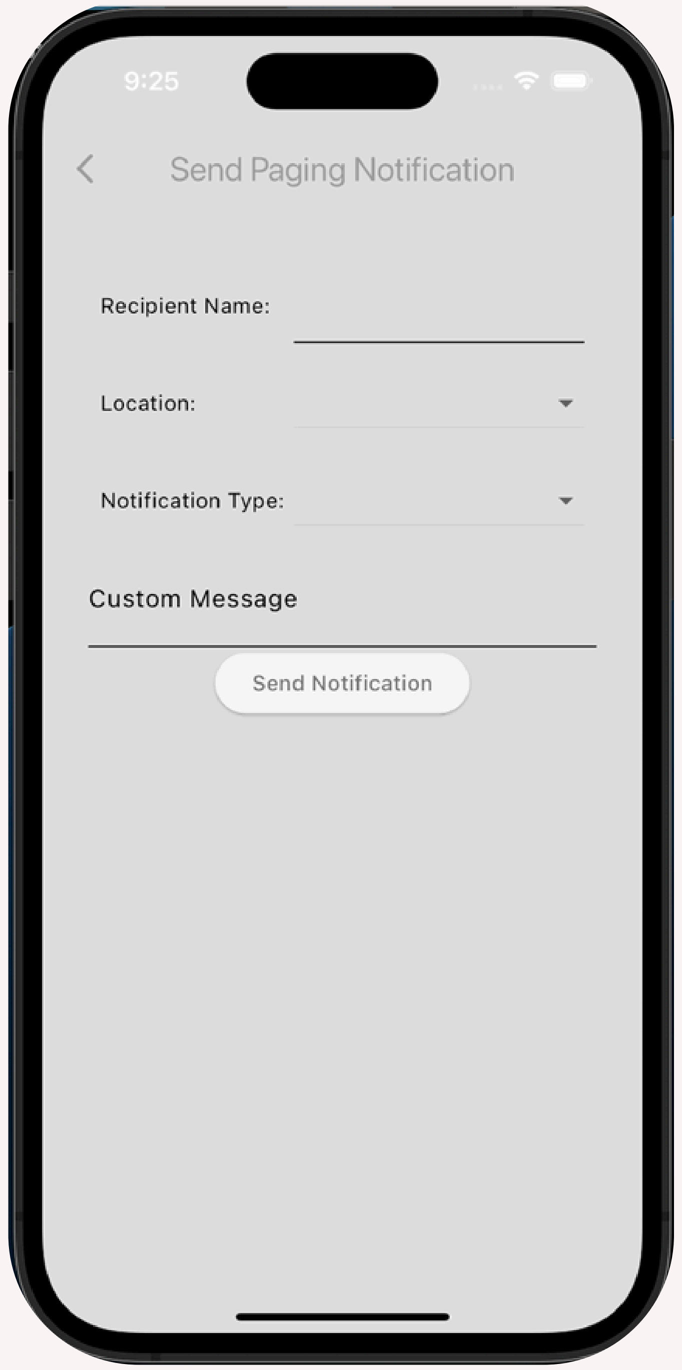


hi how are you?

I'm good how are you

hello

Paging



9:25

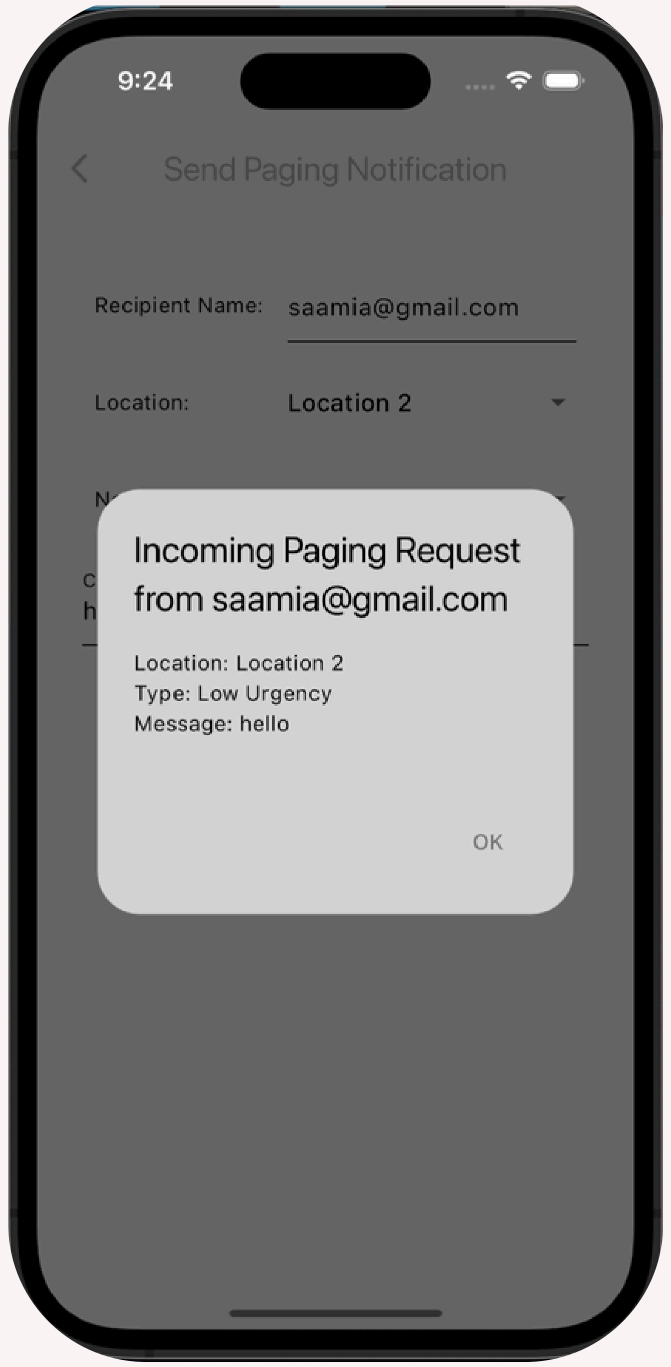
Send Paging Notification

Recipient Name:

Location:

Notification Type:

Custom Message



9:24

Send Paging Notification

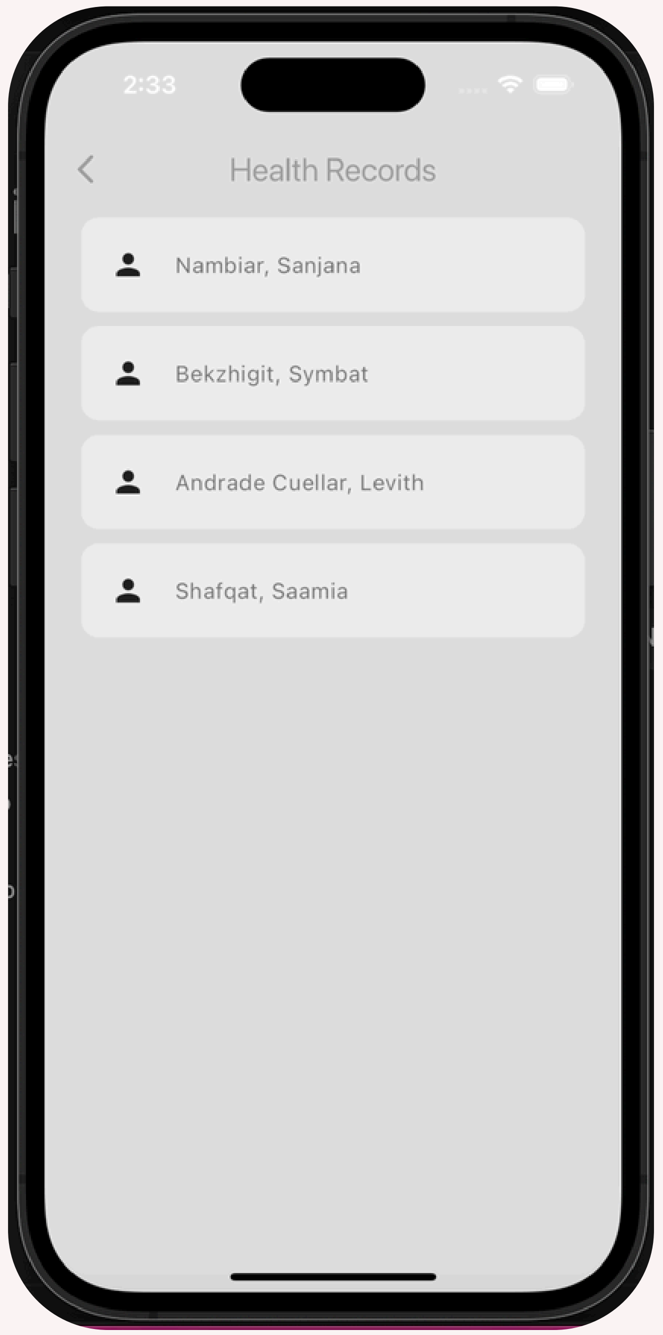
Recipient Name: saamia@gmail.com

Location: Location 2

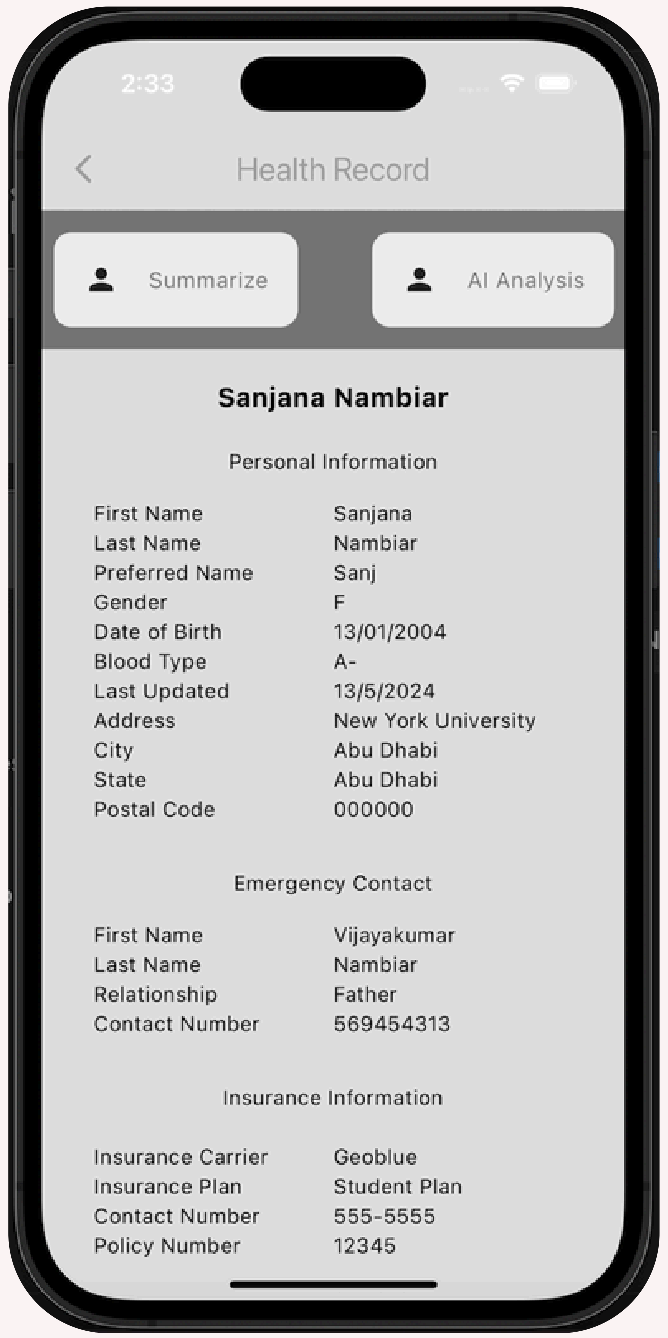
Incoming Paging Request from saamia@gmail.com

Location: Location 2
Type: Low Urgency
Message: hello

EHR Viewing + AI



- Nambiar, Sanjana
- Bekzhigit, Symbat
- Andrade Cuellar, Levith
- Shafqat, Saamia



Sanjana Nambiar

Personal Information

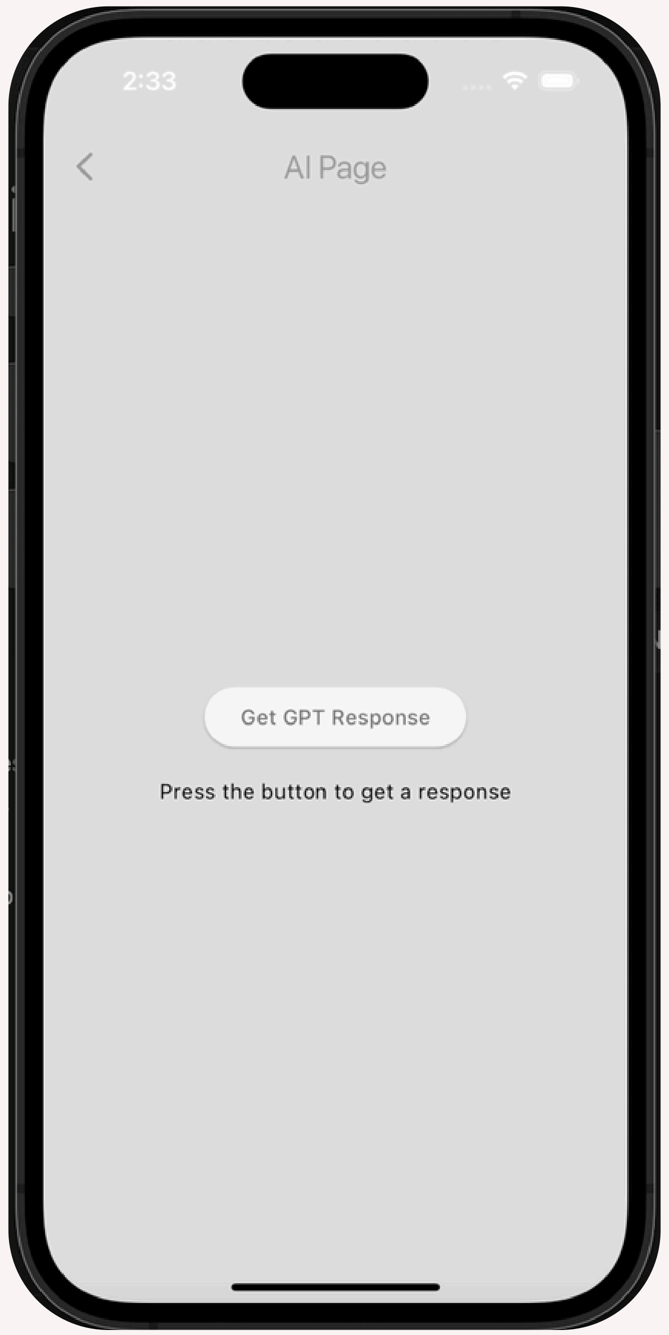
First Name	Sanjana
Last Name	Nambiar
Preferred Name	Sanj
Gender	F
Date of Birth	13/01/2004
Blood Type	A-
Last Updated	13/5/2024
Address	New York University
City	Abu Dhabi
State	Abu Dhabi
Postal Code	000000

Emergency Contact

First Name	Vijayakumar
Last Name	Nambiar
Relationship	Father
Contact Number	569454313

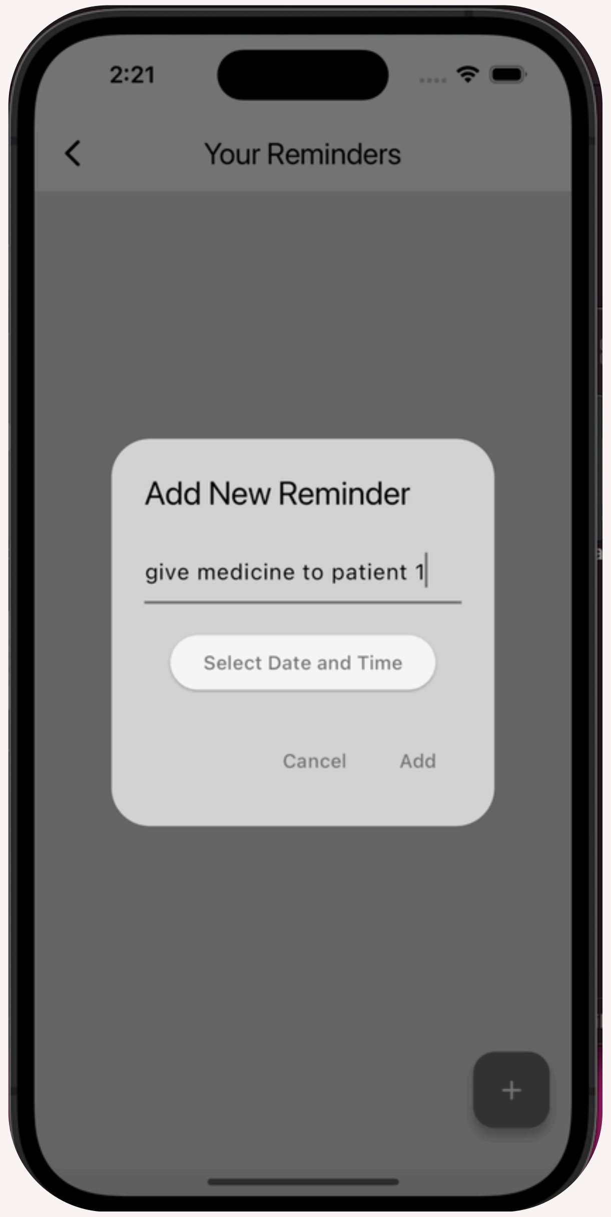
Insurance Information

Insurance Carrier	Geoblue
Insurance Plan	Student Plan
Contact Number	555-5555
Policy Number	12345



Press the button to get a response

Reminders

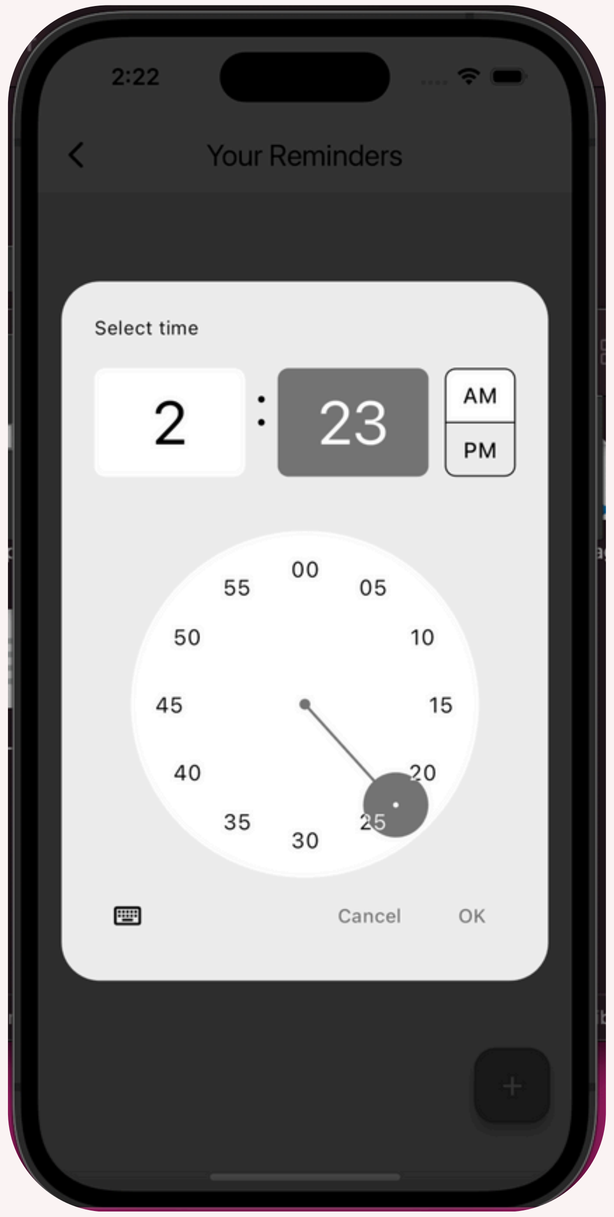


2:21

Your Reminders

Add New Reminder

give medicine to patient 1

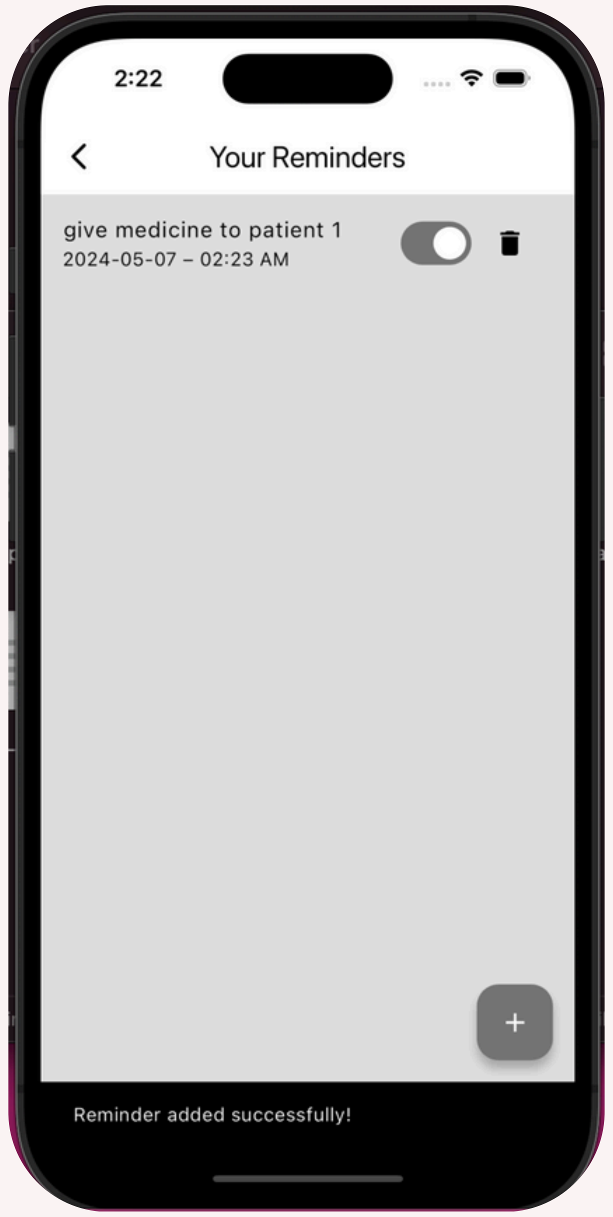


2:22

Your Reminders

Select time

2 : 23 AM PM

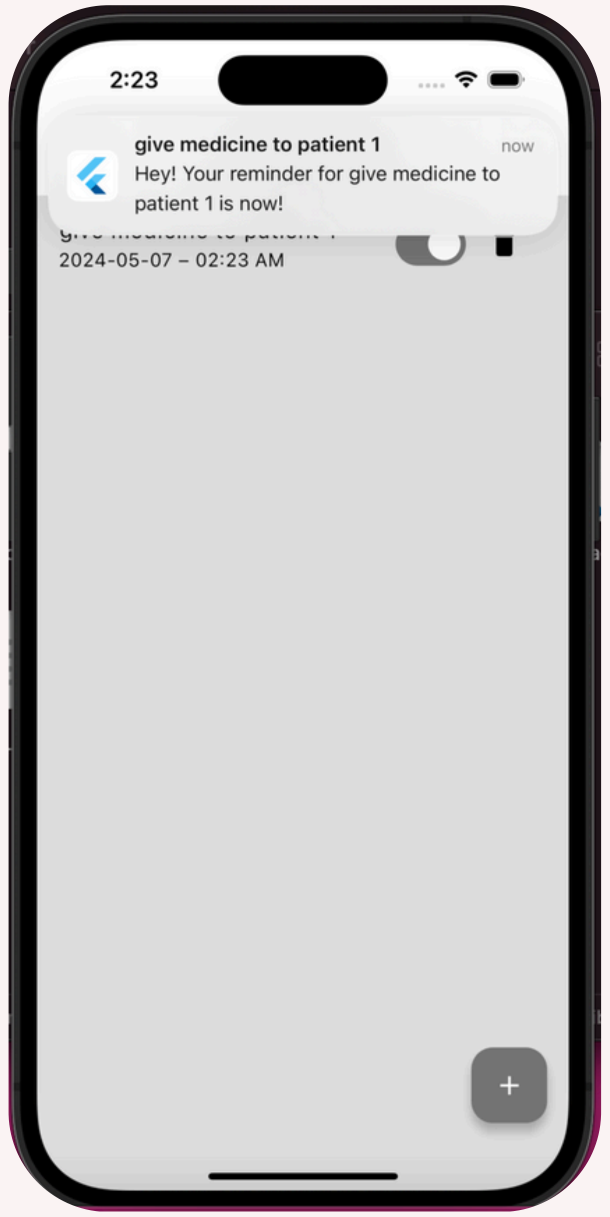


2:22

Your Reminders

give medicine to patient 1
2024-05-07 – 02:23 AM

Reminder added successfully!



2:23

give medicine to patient 1 now
Hey! Your reminder for give medicine to patient 1 is now!

2024-05-07 – 02:23 AM

Development Tools

Flutter

Firebase

VS Code

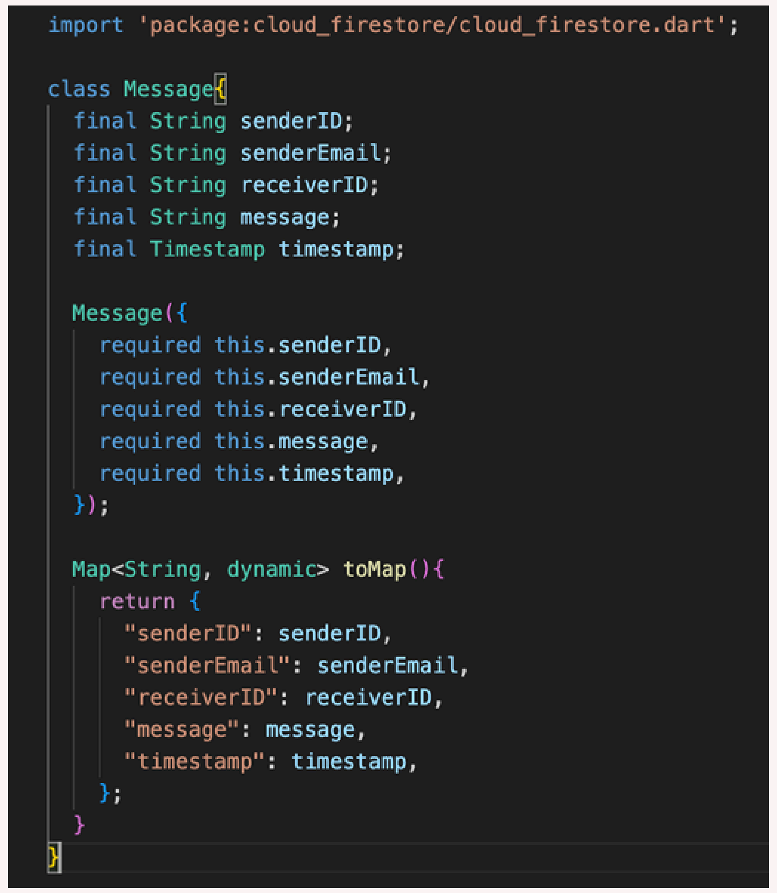
Xcode

Challenges Faced

- Pushing notifications across devices.
- Learning new technologies from scratch.
- Merging features together.

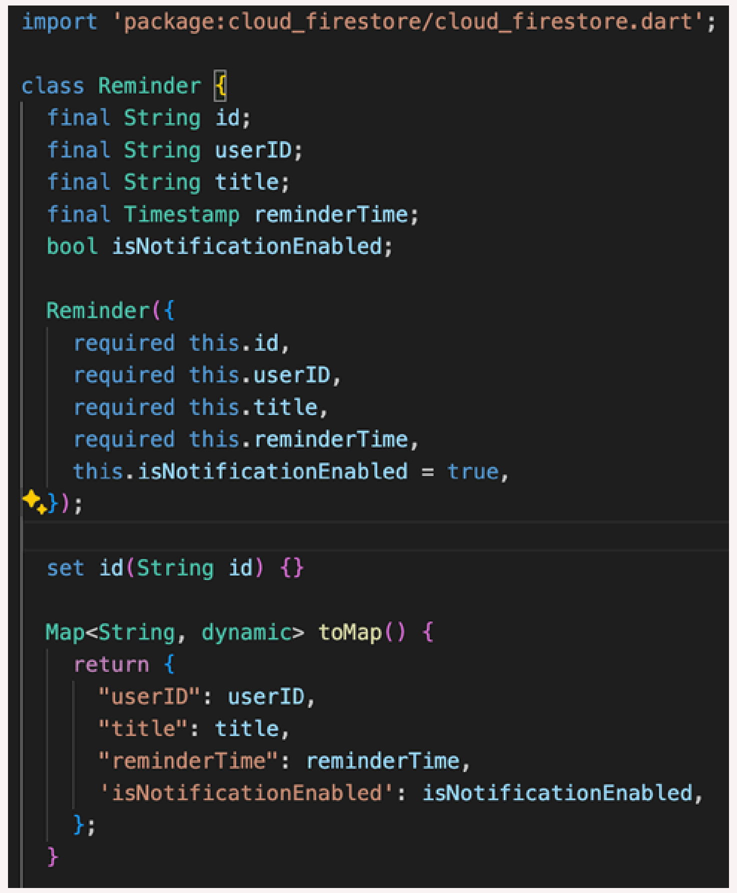
Code Highlights

connecting with the Class Diagram



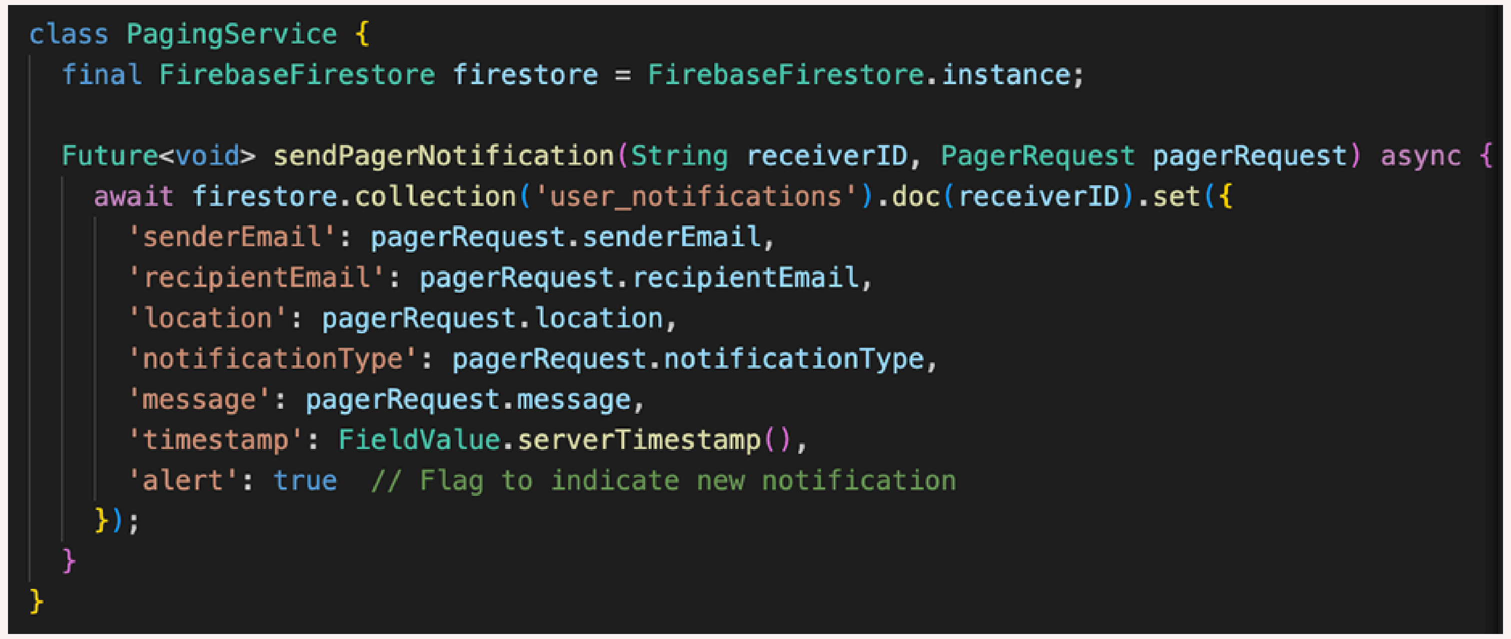
```dart
import 'package:cloud_firestore/cloud_firestore.dart';

class Message{
  final String senderID;
  final String senderEmail;
  final String receiverID;
  final String message;
  final Timestamp timestamp;

  Message({
    required this.senderID,
    required this.senderEmail,
    required this.receiverID,
    required this.message,
    required this.timestamp,
  });

  Map<String, dynamic> toMap(){
    return {
      "senderID": senderID,
      "senderEmail": senderEmail,
      "receiverID": receiverID,
      "message": message,
      "timestamp": timestamp,
    };
  }
}
```

Message Class



```dart
import 'package:cloud_firestore/cloud_firestore.dart';

class Reminder {
  final String id;
  final String userID;
  final String title;
  final Timestamp reminderTime;
  bool isNotificationEnabled;

  Reminder({
    required this.id,
    required this.userID,
    required this.title,
    required this.reminderTime,
    this.isNotificationEnabled = true,
  });

  set id(String id) {}

  Map<String, dynamic> toMap() {
    return {
      "userID": userID,
      "title": title,
      "reminderTime": reminderTime,
      'isNotificationEnabled': isNotificationEnabled,
    };
  }
```

Reminder Class



```dart
class PagingService {
  final FirebaseFirestore firestore = FirebaseFirestore.instance;

  Future<void> sendPagerNotification(String receiverID, PagerRequest pagerRequest) async {
    await firestore.collection('user_notifications').doc(receiverID).set({
      'senderEmail': pagerRequest.senderEmail,
      'recipientEmail': pagerRequest.recipientEmail,
      'location': pagerRequest.location,
      'notificationType': pagerRequest.notificationType,
      'message': pagerRequest.message,
      'timestamp': FieldValue.serverTimestamp(),
      'alert': true  // Flag to indicate new notification
    });
  }
}
```

Paging Service (Paging System)

Testing Approach

We mostly did

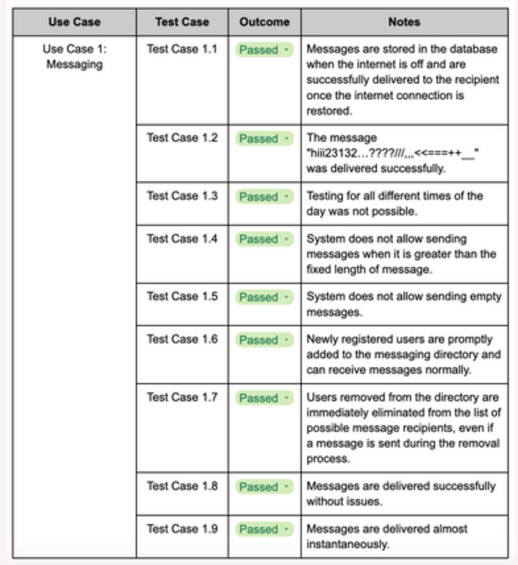
Manual Testing

but we also incorporated the use of

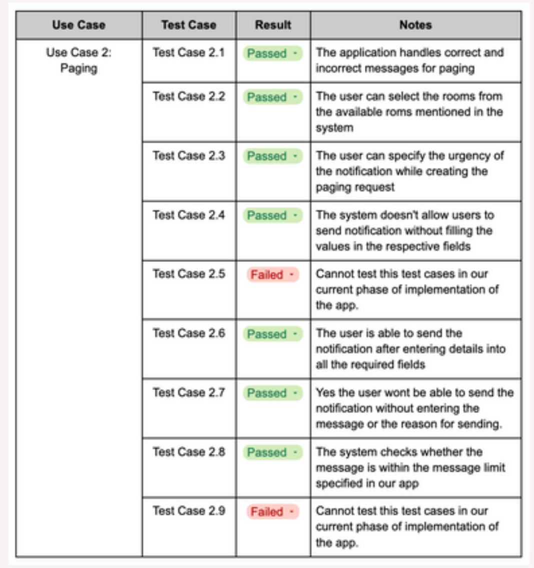
pairwiseTool

for Pairwise Testing.

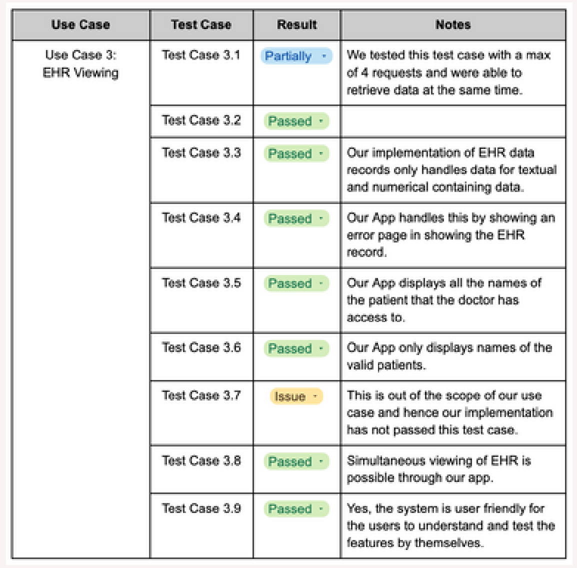
Testing Results



Use Case	Test Case	Outcome	Notes
Use Case 1: Messaging	Test Case 1.1	Passed	Messages are stored in the database when the internet is off and are successfully delivered to the recipient once the internet connection is restored.
	Test Case 1.2	Passed	The message "hiii23132…????///,,,<<===++__" was delivered successfully.
	Test Case 1.3	Passed	Testing for all different times of the day was not possible.
	Test Case 1.4	Passed	System does not allow sending messages when it is greater than the fixed length of message.
	Test Case 1.5	Passed	System does not allow sending empty messages.
	Test Case 1.6	Passed	Newly registered users are promptly added to the messaging directory and can receive messages normally.
	Test Case 1.7	Passed	Users removed from the directory are immediately eliminated from the list of possible message recipients, even if a message is sent during the removal process.
	Test Case 1.8	Passed	Messages are delivered successfully without issues.
	Test Case 1.9	Passed	Messages are delivered almost instantaneously.



Use Case	Test Case	Result	Notes
Use Case 2: Paging	Test Case 2.1	Passed	The application handles correct and incorrect messages for paging
	Test Case 2.2	Passed	The user can select the rooms from the available roms mentioned in the system
	Test Case 2.3	Passed	The user can specify the urgency of the notification while creating the paging request
	Test Case 2.4	Passed	The system doesn't allow users to send notification without filling the values in the respective fields
	Test Case 2.5	Failed	Cannot test this test cases in our current phase of implementation of the app.
	Test Case 2.6	Passed	The user is able to send the notification after entering details into all the required fields
	Test Case 2.7	Passed	Yes the user wont be able to send the notification without entering the message or the reason for sending.
	Test Case 2.8	Passed	The system checks whether the message is within the message limit specified in our app
	Test Case 2.9	Failed	Cannot test this test cases in our current phase of implementation of the app.

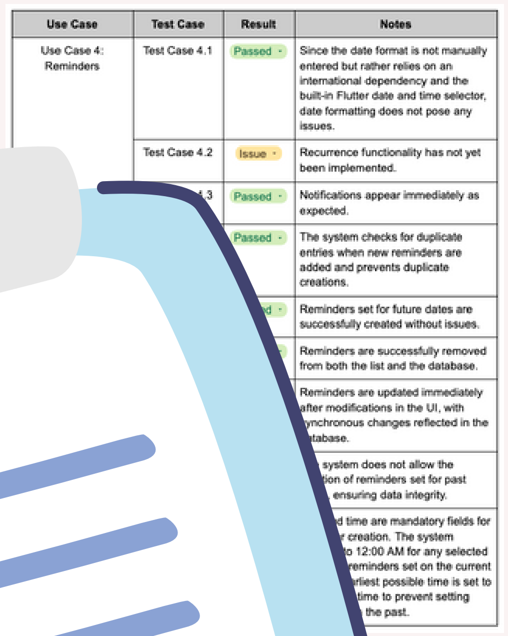


Use Case	Test Case	Result	Notes
Use Case 3: EHR Viewing	Test Case 3.1	Partially	We tested this test case with a max of 4 requests and were able to retrieve data at the same time.
	Test Case 3.2	Passed	
	Test Case 3.3	Passed	Our implementation of EHR data records only handles data for textual and numerical containing data.
	Test Case 3.4	Passed	Our App handles this by showing an error page in showing the EHR record.
	Test Case 3.5	Passed	Our App displays all the names of the patient that the doctor has access to.
	Test Case 3.6	Passed	Our App only displays names of the valid patients.
	Test Case 3.7	Issue	This is out of the scope of our use case and hence our implementation has not passed this test case.
	Test Case 3.8	Passed	Simultaneous viewing of EHR is possible through our app.
	Test Case 3.9	Passed	Yes, the system is user friendly for the users to understand and test the features by themselves.

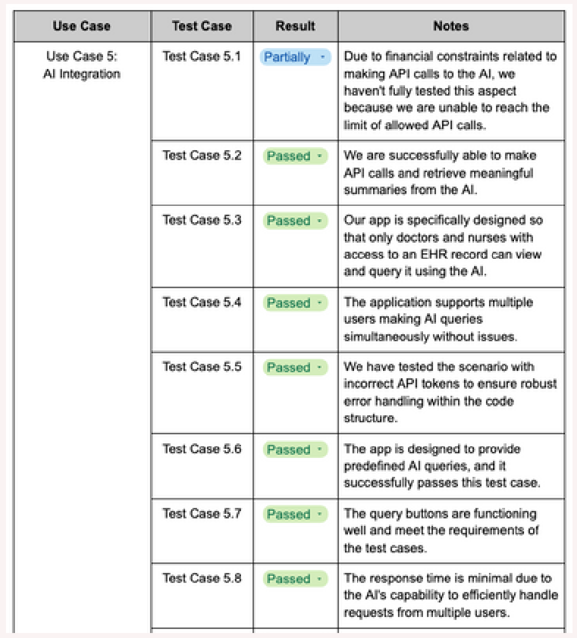
Reflections

Due to time constraints, we were unable to perform multiple ***daily test iterations*** *for each test case.*

Couldn't test the maximum API calls to AI due to budget constraints



Use Case	Test Case	Result	Notes
Use Case 4: Reminders	Test Case 4.1	Passed	Since the date format is not manually entered but rather relies on an international dependency and the built-in Flutter date and time selector, date formatting does not pose any issues.
	Test Case 4.2	Issue	Recurrence functionality has not yet been implemented.
	[illegible].3	Passed	Notifications appear immediately as expected.
	[illegible]	Passed	The system checks for duplicate entries when new reminders are added and prevents duplicate creations.
	[illegible]	[illegible]d	Reminders set for future dates are successfully created without issues.
	[illegible]	[illegible]	Reminders are successfully removed from both the list and the database.
	[illegible]	[illegible]	Reminders are updated immediately after modifications in the UI, with ynchronous changes reflected in the atabase.
	[illegible]	[illegible]	system does not allow the tion of reminders set for past ensuring data integrity.
	[illegible]	[illegible]	d time are mandatory fields for r creation. The system o 12:00 AM for any selected reminders set on the current rliest possible time is set to time to prevent setting the past.



Use Case	Test Case	Result	Notes
Use Case 5: AI Integration	Test Case 5.1	Partially	Due to financial constraints related to making API calls to the AI, we haven't fully tested this aspect because we are unable to reach the limit of allowed API calls.
	Test Case 5.2	Passed	We are successfully able to make API calls and retrieve meaningful summaries from the AI.
	Test Case 5.3	Passed	Our app is specifically designed so that only doctors and nurses with access to an EHR record can view and query it using the AI.
	Test Case 5.4	Passed	The application supports multiple users making AI queries simultaneously without issues.
	Test Case 5.5	Passed	We have tested the scenario with incorrect API tokens to ensure robust error handling within the code structure.
	Test Case 5.6	Passed	The app is designed to provide predefined AI queries, and it successfully passes this test case.
	Test Case 5.7	Passed	The query buttons are functioning well and meet the requirements of the test cases.
	Test Case 5.8	Passed	The response time is minimal due to the AI's capability to efficiently handle requests from multiple users.

Demo

Conclusion

Achievements

Centralized application that makes information easily accesible to doctors.

Lessons Learned

Room for improvement in non-functional requirements, aided by user interactions.

Thank You for Your Attention!