**p[CS673 Software Engineering** 

**Team 1 - Med Tracker**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Andrew Gieraltowski | Team leader | *AJG* | 2022/9/11 |
| Yuan Wang | Requirement Leader | *Yuan Wang* | 2022/9/11 |
| Yuan Wang | Design and Implementation Leader | *Yuan Wang* | 2022/9/11 |
| Haiyang Lu | QA leader | Haiyang Lu | 2022/9/11 |
| Andrew Gieraltowski | Configuration Leader | *AJG* | 2022/9/11 |
| Haiyang Lu | Security Leader | Haiyang Lu | 2022/9/11 |
| [Divya Thomas](mailto:divthomas22@gmail.com) | Design and Implementation Leader | *Divya Thomas* | 2022/9/11 |
| [Divya Thomas](mailto:divthomas22@gmail.com) | QA Leader | *Divya Thomas* | 2022/9/11 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 1 | Divya Thomas | 2022/9/25 | Introduction |
| 2 | Yuan Wang | 2022/9/26 | Class Diagram  UI Design |
| 3 | Divya Thomas | 2022/9/26 | Software Architecture, Database Design, Security Design, Business Logic/Key Algorithms, Design Patterns |
| 4 | Divya Thomas | 2022/10/18 | Updating all sections |

[Introduction](#_87t9hln2vjz0)

[Software Architecture](#_3ipvmjgn6clp)

[Class Diagram](#_ky60nv8suxxm)

[UI Design (if applicable)](#_7ucksmkf6rzx)

[Database Design (if applicable)](#_tcmuor4nl1kz)

[Security Design](#_x18fj36s1121)

[Business Logic and/or Key Algorithms](#_mtfbusfb0eq3)

[Design Patterns](#_9zvwkmc4luo5)

[Any Additional Topics you would like to include.](#_15tmymhipvdv)

[References](#_50ojo9i46ytq)

[Glossary](#_8n34lvocupub)

# Introduction

This document displays an overview of the design structure and implementation of the MedTracker application. While elaborating on goals and objectives discussed in the SPPP, this document will be continuously updated throughout 3 iterations as the product is created and refined.

The MedTracker software system is an Android application to allow a user to manage and organize their medications and schedules. While this application is useful to the general public, the MedTracker’s team is primarily focused on creating a tool that appeals to an elderly age demographic. With this in mind, this project is targeting the following design goals below.

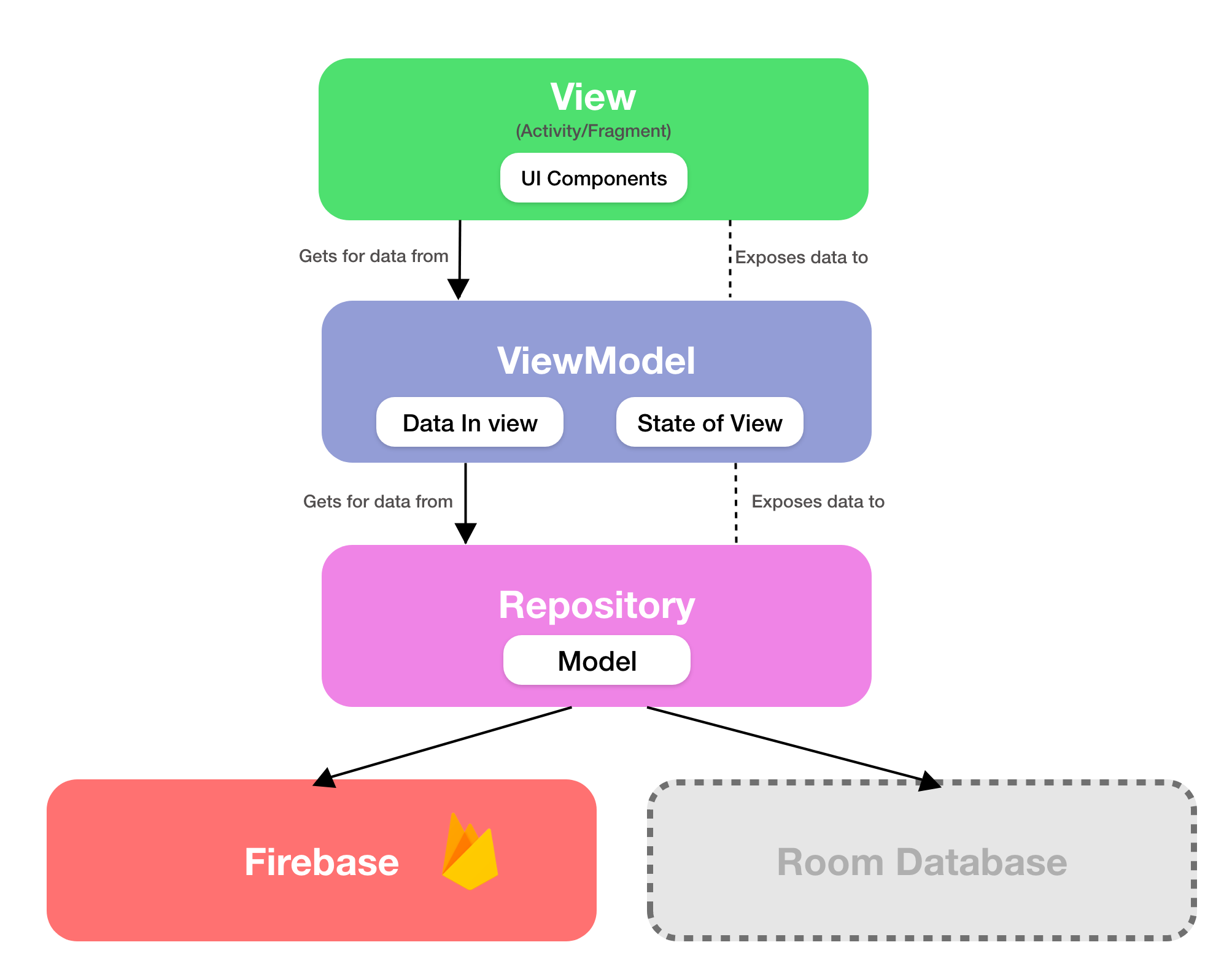
* **Simple, consistent, and organized UI**
* **No overcrowding with features**
* **Minimum steps/clicks to accomplish a task**
* **Quickly learnable**
* **Customizable**
* **Accessible to both Patients and Caretakers**

# Software Architecture

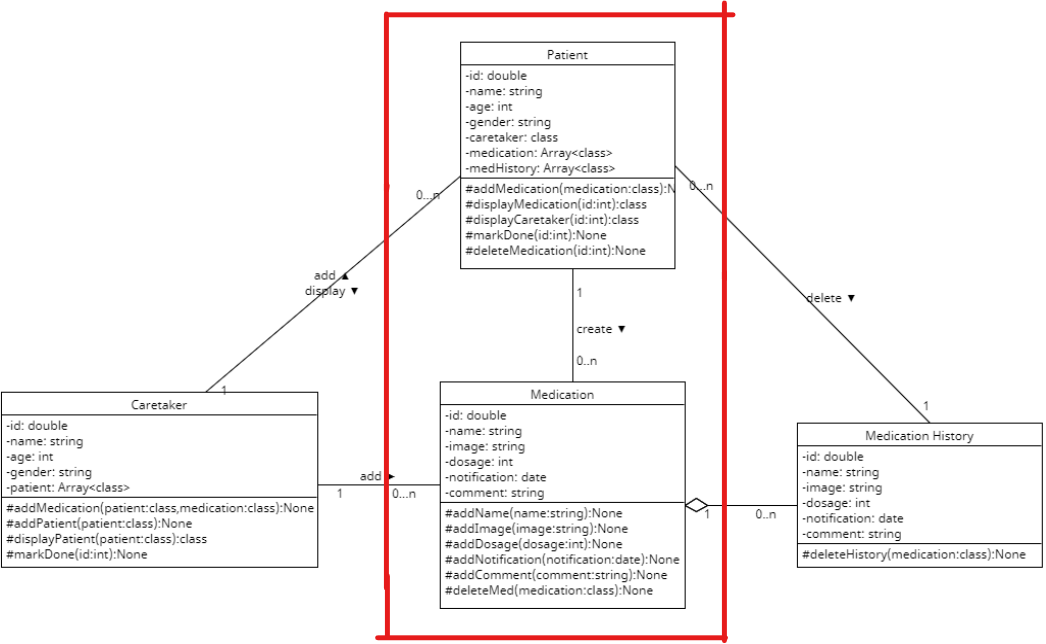
The architecture for MedTracker aims to follow a typical Model-View-ViewModel design organization. The MVVM design allows decoupling of the View and Model which are interactive through the “middle-man” Viewmodel.

The model will incorporate a Room database to store all data on patient medications. The view will have simple basic and similar configurations for each page and will be configured to easily switch between pages or fragments. Both the view and model will interact with the viewmodel layer in order to communicate with each other. Any change submitted through the view that requires access to the database will be communicated through the viewmodel methods, which will contain the logic of what to do with this data, such as changing states within the model. Data from the model will also be retrieved by the methods implemented in the viewmodel, which are called by the view.

A specific use of how this process is used in the process of saving a new medication for the patient. A patient or caretaker, can input new medication data on the “Add New Medication” page in the view. On submit, this view will then create a medication object with this information and use the addMed() method within the MedListViewModel object. This method accesses the database model and insert a new record with this information into the Medication table.



# Class Diagram



This is the class diagram showing the relationship between each components. There are four major components: Caretaker, patient, medication, and medication history. The components will be described in detail.

The caretaker component is dealing with adding patients, medications, and assign medication to history. The caretaker has its unique id, name, age, patients. The caretaker is able to add the patient using the “addPatient” function. The caretaker is also able to assign medication to specific patient using the “addMedication” function. By marking the medication as done, the caretaker will move the medication to history using the “markDone” function. Caretaker can also review the patient as a list by calling the “displayPatient” function.

The patient component is dealing with adding medication, displaying medication, displaying caretaker, and delete medication history. It is associated with caretaker since the patient is able to check his caretaker by calling the “displayCaretaker” class. The patient is associated with the medication component. Patient is able to add medication by calling the “addMedication” function. The patient is also associated with medication history component by deleting the history using “deleteMedication” function.

The medication component is dealing with adding the detailed information of the medicine, including the image, name of medicine, number of dosage, notification, and comments. It is the composition of medication history component. Without medication, there is no medication history. One medication can support multiple medication history. Both the medication and the medication history can be deleted by the patient.

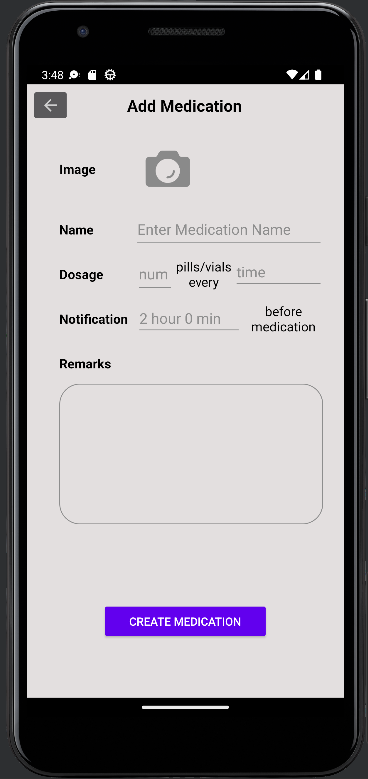
Due to the time limitations of the course, we decided to minimize the scope of the project. This, in turn, removed both the Caretaker and Medication History objects and implementation. The Patient and Medication objects and relationships persist in the final submission of the MedTracker project.

# UI Design

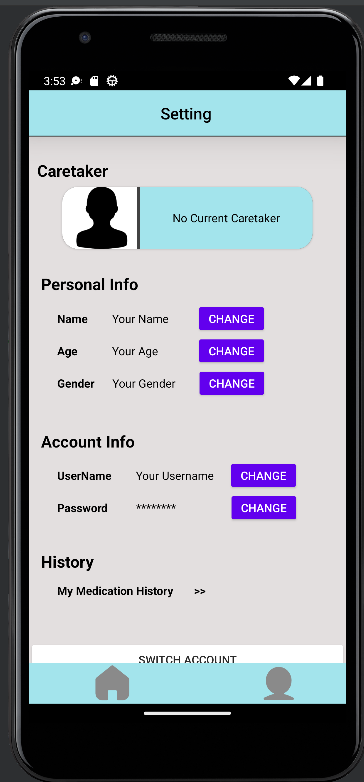
The user interface was consisting of two major parts targeting two different set of customers. There is the medication UI for the patient and the UI for the caretaker. The design strategy for the patient is to be simple and understandable. The strategy for the caretaker is to be information-detailed.



Above is the home interface. It is the main interface for the app. This is targeting the patients. In this page, the patient is able to view the medication information and the time and the number of pills he needs to take. The patient is also able to switch to the setting page by clicking the “person” image button on the bottom right corner. By clicking the “add medication” button, the patient is able to generate a new medication and it will diaplay on the home interface as a card.



This is the “add medication” interface. It is used for patients to add their own medication. The patient is able to type the medication name, number of dosages, notification and other remarks into the page. After clicking the “create medication” button, the patient is able to gernerate a new medication tag showing on the home interface. The patient is also able to create a medication tag by simply uploading a picture of that medicine.



This is the setting interface. It includes the information about the patient and the caretaker. Under this interface, the patient is able to view his personal information, including name, age, and gender; account information, including username and password; history information, including the patient’s medication history. The patient is able to change these information by clicking the “change” button on the right. The interface is scrollable, so below the medication history, there are two buttons, the “switch account” button and the “logout” button. These are buttons that designed for the patients to deal with account manipulations. The patient is able to switch to another account or logout to the current account.

# Database Design

The local Room database configured for the MedTracker application is currently composed of one main table.

The Medication table is a storage of all medication information for the patient. All medication entries contain a medication ID, which is unique and auto-incremented, the fields for name of medication, dosage amount, reminder time, and frequency of reminder will all be inputted from the UI when the patient adds the medication. This is the same data which will be displayed when the user requests to display a list of their medications. The reminder will be used when the push notifications are configured to indicate when the medication reminder will be active on the user’s UI. The frequency value will be used when the reminder needs to be updated to the next reminder time.

The database itself is a singleton database class. This was created in order to minimize the number of times a database object is expensively created. A singleton class allows only one instance to be created and consistently accessed throughout the lifecycle of the application. This way, the user is able to consistently add, edit and delete data from the same database each time, and ensures persistent data.

# Security Design

The security design for the MedTracker application hasn’t been extensively implemented or tested yet, but the intent for the application is to provide only the necessary amount of information to each user in order for them to complete their basic tasks and functionalities.

A patient account only has access to the medications, profile, and other information for their own account. They will also be able to modify only their own information. In the event that we had more time to complete this project, the additional user of a caretaker would also be considered. A caretaker, on the other hand, has access to their information, as well as the information of their registered patients. While they will be unable to edit a patient’s personal identifying information, they will be able to add and modify a patient’s medication, as long as they are listed as the patients’ caregiver.

Since this application’s data is locally stored within the user’s device, there will be no transferring of data to remote resources, which can increase the risk of data breaches. The local application is protected by the user device’s security features.

# Business Logic and/or Key Algorithm

Due to the minimization of the project scope, the pathway from the login and verification is now omitted, as well as the caretaker account, as discussed in the previous sections of this document. The current business logic is indicated by the red outline.

# Design Patterns

As discussed in the Software Architecture section of this document, the MedTracker application follows an MVVM design, with the model to encapsulate data storage, the view to render the UI, and the viewmodels to communicate between the two.

There is also a configuration of a strategy pattern with the Frequency interface and its corresponding implementations. This allows the reminder to be updated according to it’s corresponding frequency value.

In addition to this, MedTracker also implements a singleton design for the database instance. Since the database will be accessed from many different classes and processes, it would be less expensive, and more consistent and rational to have a single instance of the database that is reused, throughout the life cycle of the application. The application would also need to access the database even when not open on the foreground. This is for the reminder functionality with push notifications.

# Any Additional Topics you would like to include.

N/A

# References

* <https://yunze-li.github.io/2021/01/15/AndroidArchitecture/>

# Glossary

* UI: User Interface
* MVP: Model-View-Presenter