CS673 Software Engineering
Team X - Project Name
Software Design Document

Team Member	Role(s)	Signature	Date
Jake Stephens	Backup, QA	*Jake Stephens*	9/19/2021
Seema Palora	Team Lead,Developer	*Seema Palora*	9/20/2021
Sandeep Agrawal	Design and Implementation Lead	*Sandeep Agrawal*	9/20/2021
Shepherd Liu	Configuration Leader	*Shepherd Liu*	9/27/2021
Yang Ye	Requirement leader, Security Leader.	*Yang Ye*	09/27/2021

Revision history

Version	**Author**	**Date**	**Change**

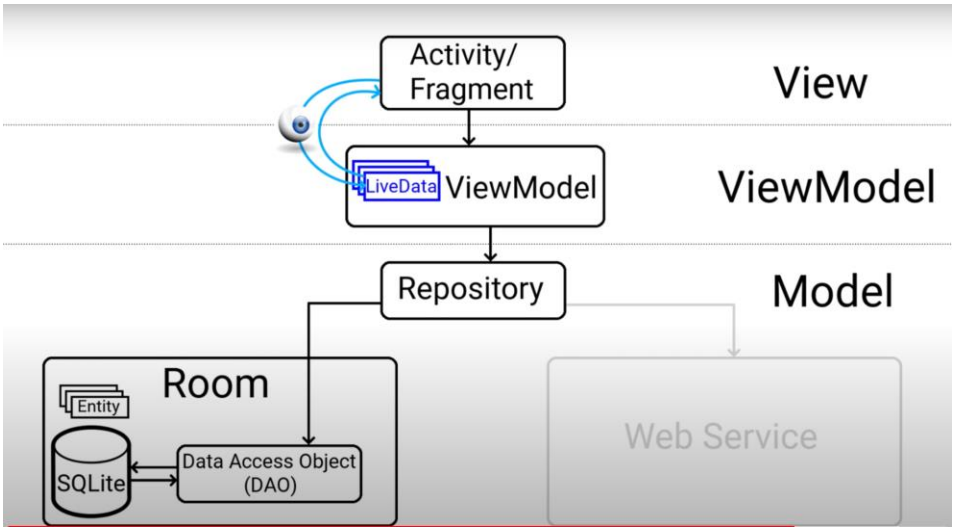
- # Introduction

Simple Health is an Android app ,developed using Java,Kotlin and Android libraries.Using local sqlite as database , will migrate to firebase for storage in future. This document gives detailed explanation on application architecture, database design, security design , UI wires and class information.

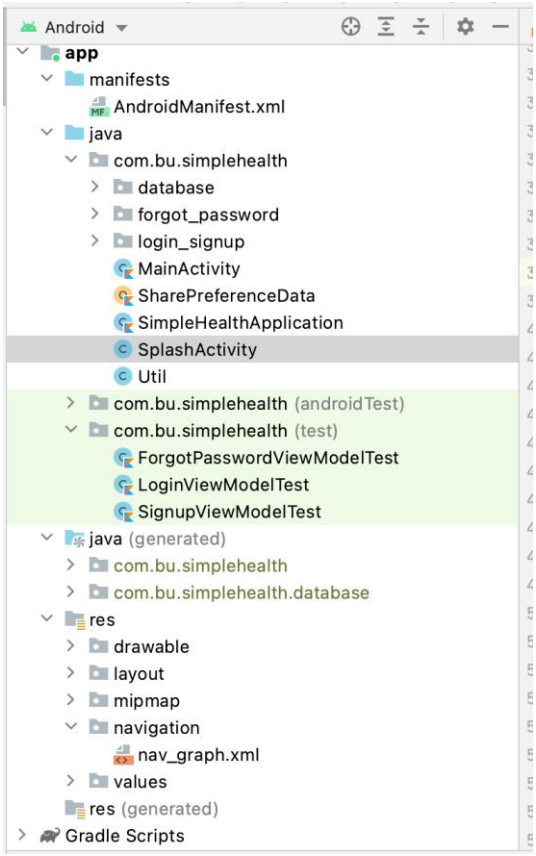
- # Software Architecture:

Application follows MVVM pattern with Room database. We are following Single activity design, which will have One Activity to host multiple Fragments for different screens. Jet Pack libraries are used for handling fragment navigation .

- **Model:** data source, model classes, repository.
- **View:** UI code (Activity, Fragment), XML. It sends the user action to the ViewModel but does **not** get the response back directly. To get the response, it has to subscribe to the observables which ViewModel exposes to it.
- **ViewModel:** Connection between the View and Model. ViewModel does not hold any kind of reference to the View.



Project Structure(This will get updated in every iteration as new classes, package will be introduced)



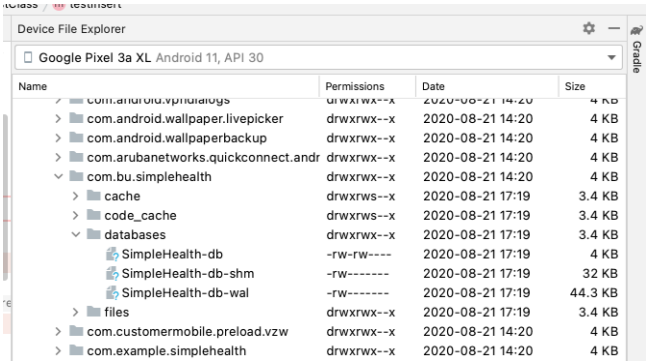
- ## Components :

  We are using following UI components:

  - AppCompatTextView
  - AppCompatButton
  - AppCompatEditText
  - Toolbar
  - AppCompatImageView
  - AppCompatCheckBox

- ## Database Design :

  Using Room local database to persist the data.



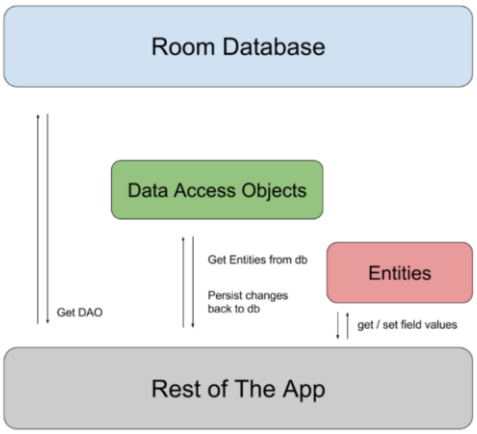
Dependencies added to the build.gradle:

```
def room_version = "2.3.0"

implementation "androidx.room:room-runtime:$room_version"
annotationProcessor "androidx.room:room-compiler:$room_version"
```

Components:

1. **Data Entity** :Represent tables in app's database
2. **DAO** :Provide methods to query, update, insert, and delete data in the database.
3. **Database class** :Holds the database and serves as the main access point for the underlying connection to your app's persisted data.



- # Security Design:

1. Password is hashed using SHA with SALT before storing to the database
2. Shared preference are defined as private
3. NPI(Non public personal info) are not logged
4. If time permits planning to add Biometric login
5. Comprehensive testing program throughout development lifecycle
6. Application-defined permissions to control application data on a per-app basis
7. User input validations performed.

Future Security Improvements:

- Since the Simple Health application stores sensitive health data for its users, it is important to have the proper security in place for storing the health data. PHI (Personal Health Information) is highly sensitive data that needs to be kept safe to ensure our customers trust in our application. That is why all health related data stored in the database for this app will be encrypted with RSA private/public keys to ensure that the data is kept safe.

Design Patterns:

Design patterns used in your software system, as of now these 2 patterns, however we may introduce some more in upcoming iterations.

- MVVM :

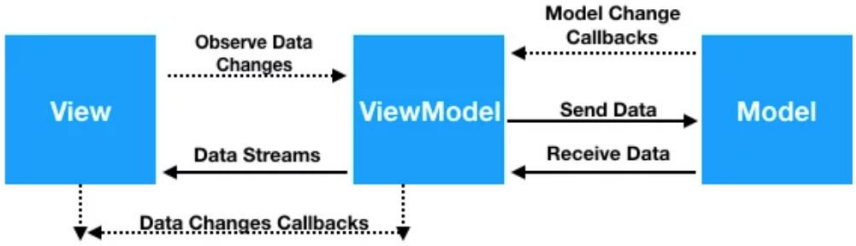


Image source:https://www.journaldev.com/20292/android-mvvm-design-pattern

Separation of business logic from the UI code. The architecture prevents data loss during configuration changes and ensures that all dependencies are present before using MVVM, thereby helping to prevent runtime errors.

- Observer Pattern(Live data usage)

Key Algorithms :

In this section, you shall describe any key algorithms used in your software system, either in terms of pseudocode or flowchart.

- For security purposes, we will be hashing the passwords before storing them in the database. This algorithm will hash the password using SHA-256 using salt for a more secure system.
- For the exercise notifications feature of SimpleHealth, we will use a randomizer algorithm to help trigger notifications to the user when they should receive instruction to do a certain exercise. The algorithm will be taking into consideration 2 parameters. The first parameter is the type of exercise, whether it is a core exercise, a stretching exercise, etc. The second parameter is the frequency. This will determine the time that the user will be notified. The frequency variable has 4

possible values on the UI side, either *Very Rarely*, *Rarely*, *Often*, or *Very Often*. The value of this parameter will be translated into a number, which is used to calculate how often the user is notified.

- Here is an example of pseudocode for this algorithm (code snippet will be provided once code is written and implemented in the app)

```
method exerciseInterval(exerciseType, frequency)
    - select group of exercises based on exerciseType
    - from group of exercises, pick 1 at random
    - based on frequency, a wait time will be determined
    - application will open a thread and sleep
    - application will send push notification
```

- Some other algorithms that will be used are our verification algorithms. These algorithms' purpose will be to validate that the fields entered in by the user follow our standard. For example, when the user enters their date of birth on the signup screen we will validate that the text is in the format MM/DD/YYYY. For this we will use a regular expression to validate the field is in the right format. Similarly for the password field, we will have an algorithm to make sure the password contains at least one capital letter and at least one number. The pseudo code for these algorithms are as follows:

Password Validation:

```
method validatePassoword(password):
    if password length is > 10
    and
    if password contains number (0-9)
    and
    if password contains capitol letter
        return True
    else
        return False
```

Date of Birth Validation:

```
method validateDateOfBirth(dob):
    check dob is in format:
        MM/DD/YYYY
    Regex for this:
        ^(0[1-9]|1[012])[- /.] (0[1-9]|[12][0-9]|3[01])[- /.] (19|20)\d\d$
               1                          2                      3

    Part 1 is for the month.
    Part 2 is for the day.
    Part 3 is for the year.

    If dob matches regex, return True
    else return False
```

- ## UI Design :

Layout files(XML) are defined for each app screen.Using android provided UI components and to build the graphical user interface.

UI Components:

- Layouts(Linear,Relative,Constraint)
- View and Viewgroups(RadioButton, Checkbox,Spinner…)
- Material Design Components
- (Cardview,AppCompatTextView,AppCompatEditText
- ,AppCompatImageView)
- App Toolbar
- Effective navigation between fragments through jetpack library
- Custom Themes and styles

Layout performance improved :

- Reusing the layout with </include>
- Optimized layout hierarchies
- Use of Recyclerview

Wires:

https://drive.google.com/file/d/1WVCIpF4LtM3KgrWCScZeOZcDwDqAKLmS/view?usp=sharing

- ## Classes and Methods :Seema/Sandeep : Will add this section in upcoming iteration

Please provide a link to your application API document which should be automatically generated.

We will be auto generating the documentation from the project source code in the next iteration. There will be a story in PivotalTracker to capture this work.

- ## References

- ## Glossary