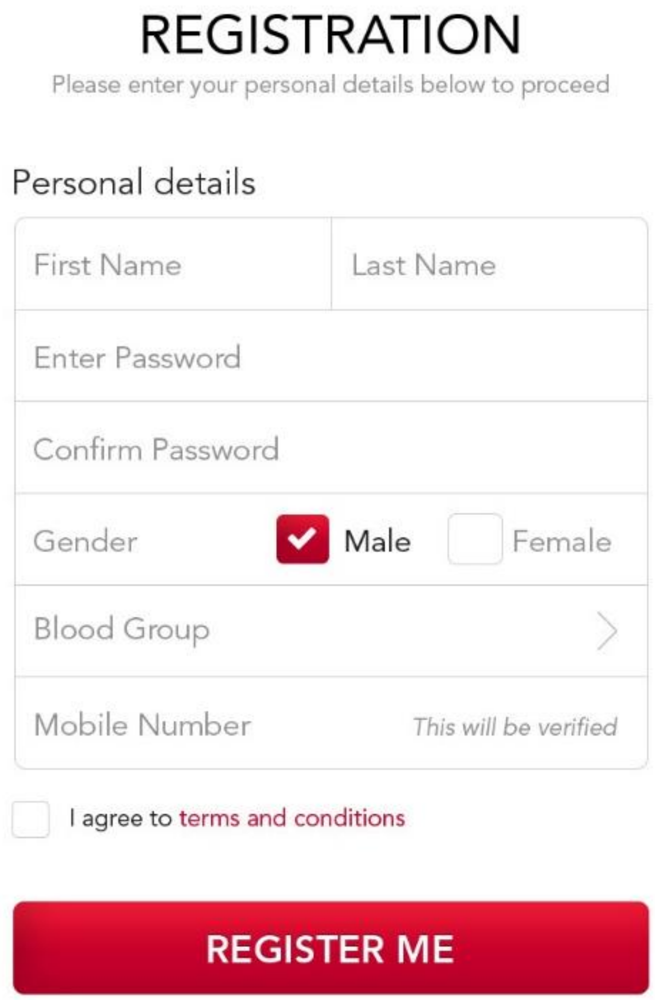
Assignment 2

Rayana Sri Sai Santhosh – 20BIT0185

Blood Bank App



REGISTRATION

Please enter your personal details below to proceed

Personal details

First Name | Last Name

Enter Password

Confirm Password

Gender ☑ Male □ Female

Blood Group >

Mobile Number *This will be verified*

□ I agree to terms and conditions

REGISTER ME

Source Code:

```
package com.mczuba.blooddonor.data.models

import androidx.room.Entity
import androidx.room.ForeignKey
import androidx.room.PrimaryKey
import java.util.*

@Entity(tableName = "donation_table")
data class Donation (
    @PrimaryKey(autoGenerate = true) var donationId: Int,
    @ForeignKey
        (entity = User::class,
        parentColumns = ["userId"],
        childColumns = ["donorId"],
        onDelete = ForeignKey.CASCADE
    )    var donorId: Int,
    var type: DonationType,
    var amount: Int,
    var date: Date,
    var location: String,
    var note: String,
    var arm: ArmType,
    var hemoglobin: Float?,
    var duration: Float?,
    var diastolic: Int?,
    var systolic: Int?
) {
    enum class DonationType {
        WHOLE,
        PLASMA,
        PLATELETS,
        DISQUALIFIED
    }

    enum class ArmType {
        LEFT,
        RIGHT,
        UNKNOWN
    }
}


package com.mczuba.blooddonorcompanion.data.models

import androidx.room.*

@Dao
interface ScheduleDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun addSchedule(schedule: Schedule)
    @Delete
    suspend fun removeSchedule(schedule: Schedule)

    @Update
    suspend fun updateSchedule(schedule: Schedule)

    @Query("SELECT * FROM schedule_table WHERE userId = :userId
ORDER BY date ASC")
    suspend fun readAllUserScheduled(userId: Int) :
List<Schedule>
}
package com.mczuba.blooddonorcompanion.data

import com.mczuba.blooddonorcompanion.data.models.Donation
import java.util.*

data class DonationSummary(
    var type: Donation.DonationType?,
    val amount: Int,
    val newest: Date?,
    val averageHemoglobin: Float?,
    val averageDuration: Float?
) {
    companion object Factory {
        @JvmStatic
        fun getEmptyDonationSummary(type: Donation.DonationType?
= null) =
            DonationSummary(type, 0, null, 0f, 0f)

        @JvmStatic
        fun getTotalSummary(donations: List<Donation>):
DonationSummary {
            var amount = 0
            var count = 0
            var countH = 0
            var countD = 0
            var hemoglobin = 0f
            var duration = 0f

            donations.forEach { donation ->
                amount += donation.amount
                count++

                donation.hemoglobin?.let {
                    hemoglobin += it
                    countH++
                }

                donation.duration?.let {
                    duration += it
                    countD++
                }
            }

            if (countH > 0)
                hemoglobin /= countH
            if (countD > 0)
                duration /= countD

            return DonationSummary(
                null,
                amount,
                donations.maxBy { it -> it.date }?.date,
                hemoglobin,
                duration
            )
        }

        @JvmStatic
        fun getTotalSummarySinceDate(donations: List<Donation>,
date: Date): DonationSummary {
            val filteredDonations = donations.filter { it ->
it.date.after(date) }

            return getTotalSummary(filteredDonations)
        }

        @JvmStatic
        fun getTypeSummary(
            donations: List<Donation>,
            type: Donation.DonationType
        ): DonationSummary {
            val filteredDonations = donations.filter { it ->
it.type == type }

            return getTotalSummary(filteredDonations)
        }

        @JvmStatic
        fun getTypeSummarySinceDate(
            donations: List<Donation>,
            type: Donation.DonationType,
            date: Date
        ): DonationSummary {
            val filteredDonations = donations.filter { it ->
it.date.after(date) }
            val filteredDonations2 = filteredDonations.filter {
it -> it.type == type }

            return getTotalSummary(filteredDonations2)
        }
    }
}
package com.mczuba.blooddonorcompanion.vm.records
```

```
import android.content.Context
import android.view.View
import com.mczuba.blooddonorcompanion.R
import com.mczuba.blooddonorcompanion.data.models.Donation
import java.text.SimpleDateFormat

class DonationData(val context: Context, val donation: Donation,
val header: Boolean = false)
{
    fun getDonationID() = donation.donationId
    fun getType() = donation.run{
        when(type){
            Donation.DonationType.WHOLE ->
context.resources.getStringArray(R.array.donation_array)[0]
            Donation.DonationType.PLASMA ->
context.resources.getStringArray(R.array.donation_array)[1]
            Donation.DonationType.PLATELETS ->
context.resources.getStringArray(R.array.donation_array)[2]
            Donation.DonationType.DISQUALIFIED ->
context.resources.getStringArray(R.array.donation_array)[3]
            else -> context.getString(R.string.donation_unknown)
        }
    }
    fun getAmount() = context.getString(R.string.format_number,
donation.amount)
    fun getArmShort() = donation.run{
        when(arm){
            Donation.ArmType.RIGHT ->
context.resources.getStringArray(R.array.arm_array_short)[0]
            Donation.ArmType.LEFT ->
context.resources.getStringArray(R.array.arm_array_short)[1]
            else ->
context.resources.getStringArray(R.array.arm_array_short)[2]
        }
    }
    fun getArm() = donation.run{
        when(arm){
            Donation.ArmType.RIGHT ->
context.resources.getStringArray(R.array.arm_array)[0]
            Donation.ArmType.LEFT ->
context.resources.getStringArray(R.array.arm_array)[1]
            else ->
context.resources.getStringArray(R.array.arm_array)[2]
        }
    }
    fun getLocation() = donation.location
    fun getYear() =
SimpleDateFormat("yyyy").format(donation.date)
    fun getDate() = SimpleDateFormat("EE, d MMM
yyy").format(donation.date)
    fun getAmountVisibility() = if (donation.amount>0)
View.VISIBLE else View.GONE
    fun getTypeColor() = when(donation.type) {
        Donation.DonationType.PLATELETS ->
context.resources.getColor(R.color.colorPrimaryDark)
        Donation.DonationType.WHOLE ->
context.resources.getColor(R.color.colorPrimary)
        Donation.DonationType.PLASMA ->
context.resources.getColor(R.color.colorPrimaryLight)
        Donation.DonationType.DISQUALIFIED ->
context.resources.getColor(R.color.colorBlack)
    }
    fun getDuration() = donation.duration?.run {
        context.getString(R.string.format_minutes, this)
    } ?: "-"

    fun getHemoglobin() = donation.hemoglobin?.run {
        context.getString(R.string.format_decimal, this)
    } ?: "-"

    fun getBloodPressure() : String {
        if(donation.diastolic==null || donation.systolic==null)
            return "-"
        else
            return context.getString(R.string.format_pressure,
donation.diastolic, donation.systolic)
    }
        fun getNote() = donation.note
}
```