**Admin.kt:**

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "admins")

data class Admin(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val password: String,

val email: String

)

**Patient.kt (login)**

import androidx.room.Entity

import androidx.room.PrimaryKey

import java.time.LocalDate

import java.time.format.DateTimeFormatter

@Entity(tableName = "patients")

data class Patient(

@PrimaryKey(autoGenerate = true) val patient\_id: Int = 0,

val password: String,

val email: String,

val registrationMonth: String = getCurrentMonth() // Auto-generated month

) {

companion object {

fun getCurrentMonth(): String {

val currentDate = LocalDate.now()

val formatter = DateTimeFormatter.ofPattern("MMMM yyyy") // Example: "February 2025"

return currentDate.format(formatter)

}

}

}

**Doctor.kt (login)**

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "doctors")

data class Doctor(

@PrimaryKey(autoGenerate = true) val doctor\_id: Int = 0,

val password: String,

val email: String,

val registrationMonth: String = getCurrentMonth() // Auto-generated month

) {

companion object {

fun getCurrentMonth(): String {

val currentDate = LocalDate.now()

val formatter = DateTimeFormatter.ofPattern("MMMM yyyy") // Example: "February 2025"

return currentDate.format(formatter)

}

}

}

**PatientProfile.kt :**  
import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "patient\_profile")

data class PatientProfile(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val name: String,

val smokingHabit: Boolean, // true = Yes, false = No

val waistCircumference: WaistSize,

val pregnancyCount: PregnancyCount,

val physicalActivity: PhysicalActivity,

val familyDiabetesHistory: FamilyDiabetesHistory,

val hdlValue: Float,

val hasPCOS: PCOSStatus,

val diabetesType: DiabetesType,

val age: Int,

val weight: Float, // in kg

val height: Float, // in cm

val sugarLevel: Float,

val gender: Gender

) {

init {

require(hdlValue in 20.0..150.0) { "HDL value must be between 20 and 150" }

}

}

enum class WaistSize {

LESS\_THAN\_31\_5, BETWEEN\_31\_5\_AND\_35, MORE\_THAN\_35

}

enum class PregnancyCount {

ZERO, ONE, TWO, THREE, MORE\_THAN\_THREE

}

enum class PhysicalActivity {

REGULAR\_EXERCISE\_AND\_STRENUOUS\_WORK,

REGULAR\_EXERCISE\_OR\_STRENUOUS\_WORK,

NO\_EXERCISE\_AND\_SEDENTARY\_WORK

}

enum class FamilyDiabetesHistory {

NO\_FAMILY\_HISTORY, ONE\_PARENT\_HAS\_DIABETES, BOTH\_PARENTS\_HAVE\_DIABETES

}

enum class PCOSStatus {

YES, NO, UNKNOWN

}

enum class DiabetesType {

TYPE\_1, TYPE\_2

}

enum class Gender {

MALE, FEMALE, OTHER

}  
  
 **DoctorProfile.kt :**import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "doctor\_profile")

data class DoctorProfile(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val name: String,

val age: Int,

val gender: Gender,

val experience: Int, // Years of experience

val specialization: String,

val practiceLicense: String, // License number

val consultancyFees: Float, // Fees in local currency

val upiId: String // UPI ID for payments

)

enum class Gender {

MALE, FEMALE, OTHER

}

**Admin side functionality🡪admin\_functionality.kt:**  
import androidx.room.Embedded

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "doctor\_requests")

data class DoctorRequest(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

@Embedded val doctorProfile: DoctorProfile,

val status: RequestStatus // PENDING, ACCEPTED, REJECTED

)

enum class RequestStatus {

PENDING, ACCEPTED, REJECTED

}

@Entity(tableName = "contributing\_doctors")

data class ContributingDoctor(

@PrimaryKey val doctorId: Int,

@Embedded val doctorProfile: DoctorProfile

)

@Entity(tableName = "doctor\_complaints")

data class DoctorComplaint(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val doctorId: Int,

val doctorName: String,

val complaint: String

)

@Entity(tableName = "doctor\_profile")

data class DoctorProfile(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val name: String,

val age: Int,

val gender: Gender,

val experience: Int, // Years of experience

val specialization: String,

val practiceLicense: String, // License number

val consultancyFees: Float, // Fees in local currency

val upiId: String // UPI ID for payments

)

enum class Gender {

MALE, FEMALE, OTHER

}

**ReportHistory.kt:**

import androidx.room.Entity

import androidx.room.ForeignKey

import androidx.room.PrimaryKey

@Entity(

tableName = "report\_history",

foreignKeys = [

ForeignKey(entity = PatientProfile::class, parentColumns = ["id"], childColumns = ["patientId"])

]

)

data class ReportHistory(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val patientId: Int, // References PatientProfile

val reportUrl: String, // Path to the generated PDF report

val generatedDate: Long // Timestamp for sorting/filtering

)

**Medication.kt(medication schedule)**

import androidx.room.Entity

import androidx.room.ForeignKey

import androidx.room.PrimaryKey

import java.text.SimpleDateFormat

import java.util.\*

@Entity(

tableName = "medications",

foreignKeys = [

ForeignKey(entity = PatientProfile::class, parentColumns = ["id"], childColumns = ["patientId"]),

ForeignKey(entity = DoctorProfile::class, parentColumns = ["id"], childColumns = ["doctorId"])

]

)

data class Medication(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val patientId: Int, // Reference to PatientProfile

val doctorId: Int, // Reference to DoctorProfile

val medicineName: String,

val dosage: String,

val timing: MedicationTiming, // Enum for "morning, afternoon, evening"

val prescribedDate: String = getCurrentDate() // Auto-generated readable date

)

enum class MedicationTiming {

MORNING, AFTERNOON, EVENING

}

// Function to get the current date in a readable format

fun getCurrentDate(): String {

val sdf = SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault())

return sdf.format(Date())

}

**VisitHistory.kt:**

import androidx.room.Entity

import androidx.room.ForeignKey

import androidx.room.PrimaryKey

import java.text.SimpleDateFormat

import java.util.\*

@Entity(

tableName = "visit\_history",

foreignKeys = [

ForeignKey(entity = PatientProfile::class, parentColumns = ["id"], childColumns = ["patientId"]),

ForeignKey(entity = DoctorProfile::class, parentColumns = ["id"], childColumns = ["doctorId"])

]

)

data class VisitHistory(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val patientId: Int, // Reference to PatientProfile

val doctorId: Int, // Reference to DoctorProfile

val visitDate: String = getCurrentDate(), // Auto-generated readable date

val prescribedMedicines: String // Comma-separated medicine names

)

// Function to get the current date in a readable format

fun getCurrentDate(): String {

val sdf = SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault())

return sdf.format(Date())

}

// Function to convert a list of medicines to a comma-separated string

fun formatMedicineList(medicines: List<String>): String {

return medicines.joinToString(", ") // Converts list to "Med1, Med2, Med3"

}

**Appointment.kt: (patient side my appointments)**

import androidx.room.\*

import java.time.LocalDate

import java.time.LocalTime

@Entity(

tableName = "appointments",

foreignKeys = [

ForeignKey(entity = PatientProfile::class, parentColumns = ["id"], childColumns = ["patientId"]),

ForeignKey(entity = DoctorProfile::class, parentColumns = ["id"], childColumns = ["doctorId"])

]

)

data class Appointment(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val patientId: Int,

val doctorId: Int,

val date: LocalDate, // Using LocalDate instead of String

val time: LocalTime // Using LocalTime instead of String

)

// TypeConverters to store LocalDate and LocalTime as String in Room database

class Converters {

@TypeConverter

fun fromLocalDate(date: LocalDate): String {

return date.toString() // Stores date as "YYYY-MM-DD"

}

@TypeConverter

fun toLocalDate(dateString: String): LocalDate {

return LocalDate.parse(dateString)

}

@TypeConverter

fun fromLocalTime(time: LocalTime): String {

return time.toString() // Stores time as "HH:mm:ss"

}

@TypeConverter

fun toLocalTime(timeString: String): LocalTime {

return LocalTime.parse(timeString)

}

}

**PatientStory.kt:**

import androidx.room.Entity

import androidx.room.PrimaryKey

import androidx.room.ForeignKey

import java.time.LocalDateTime

@Entity(tableName = "patient\_stories")

data class PatientStory(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val patientId: Int,

val doctorId: Int,

val comment: String, // Patient’s review or feedback

val postedDate: LocalDateTime, // Timestamp of when the comment was posted

@ForeignKey(entity = Patient::class, parentColumns = ["id"], childColumns = ["patientId"])

val patientReference: Int,

@ForeignKey(entity = Doctor::class, parentColumns = ["id"], childColumns = ["doctorId"])

val doctorReference: Int

)

**DoctorAppointment.kt(pinned case+progress chart+cases(3 types)):**

package com.example.yourapp.model

import androidx.room.Entity

import androidx.room.PrimaryKey

import androidx.room.TypeConverters

import java.time.YearMonth

// Enum for Case Types

enum class CaseType {

SCHEDULED,

ONGOING,

DONE

}

// Room Entity for Doctor Appointments

@Entity(tableName = "doctor\_appointments")

@TypeConverters(Converters::class) // Register TypeConverters

data class DoctorAppointment(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val doctorId: Int, // Reference to the doctor

val caseType: CaseType, // Enum: SCHEDULED, ONGOING, DONE

val progressMonth: YearMonth, // Year & Month of the appointment

val isPinned: Boolean // 1 if pinned, 0 otherwise

)

**DoctorSlots.kt (Doctor’s Slot Management):**

package com.example.yourapp.model

import androidx.room.Entity

import androidx.room.PrimaryKey

import androidx.room.ForeignKey

import androidx.room.TypeConverters

import java.time.LocalDate

import java.time.LocalTime

@Entity(tableName = "doctor\_slots",

foreignKeys = [

ForeignKey(entity = Doctor::class, parentColumns = ["id"], childColumns = ["doctorId"]),

ForeignKey(entity = Patient::class, parentColumns = ["id"], childColumns = ["patientId"])

]

)

data class DoctorSlot(

@PrimaryKey(autoGenerate = true) val id: Int = 0,

val doctorId: Int, // Reference to the doctor

val patientId: Int?, // Reference to the patient (nullable if not booked)

val slotDate: LocalDate, // Date of the slot

val slotTime: LocalTime, // Time of the slot

val isBooked: Boolean // 1 if booked, 0 if available

)