**Task 2**

**Aim**: Create Tab/Features/Facilities/Functionality in your Page as per application

**Tools:** Android Studio, Ms Word

**Description about Application:**

Note-taking applications (also called note-taking apps) allow users to: Store all notes and important information digitally, usually in a cloud-based storage system. Type, write, and draw notes on the device of choice just as one would, using a pen and paper.

When you open the app after registration/logging-in, you will find the landing page with “add notes” icon in the bottom left. Through the landing page, you can easily modify any existing notes or create new ones by pressing on the icon.

While creating a new note, the app provides the following features –

Title: to easily navigate your note

Description: the main body of the note

Date and Time scheduler

Save button

You also get an option to delete a note.

This application aims to be simple and user friendly so that maximum number of people can use this on a regular basis for their daily chores or important meetings, etc.

**Result & Discussion**

* **Code:**

import android.content.Context

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.TextView

import androidx.recyclerview.widget.RecyclerView

import com.singhvikrant.mobilecomputingtasks.R

import com.singhvikrant.mobilecomputingtasks.roomDatabase.NoteEntity

class NotesAdapter(private val context: Context, private val listener: INotesAdapter):RecyclerView.Adapter<NotesAdapter.NotesViewHolder>() {

private val notesArrayList = ArrayList<NoteEntity>()

inner class NotesViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

val title:TextView? = itemView.findViewById(R.id.title)

val notes:TextView? = itemView.findViewById(R.id.notes)

val clickable:TextView = itemView.findViewById(R.id.clickable)

}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): NotesViewHolder {

val viewHolder = NotesViewHolder(LayoutInflater.from(context).inflate(R.layout.activity\_notes\_recycler, parent, false))

viewHolder.clickable.setOnClickListener{ listener.onItemClick(notesArrayList[viewHolder.adapterPosition])

}

return viewHolder

}

override fun onBindViewHolder(holder: NotesViewHolder, position: Int) {

val currentNote = notesArrayList[position]

holder.title?.text = currentNote.title

holder.notes?.text = currentNote.note

}

override fun getItemCount(): Int {

return notesArrayList.size

}

fun updateList(newList:List<NoteEntity>){

notesArrayList.clear()

notesArrayList.addAll(newList)

notifyDataSetChanged()

}

}

interface INotesAdapter {

fun onItemClick(note: NoteEntity)

}

import android.app.DatePickerDialog

import android.app.TimePickerDialog

import android.widget.\*

import com.singhvikrant.mobilecomputingtasks.R

import com.singhvikrant.mobilecomputingtasks.ui.NoteActivity

import java.text.DecimalFormat

import java.util.\*

class DateTime(private val noteView: NoteActivity): DatePickerDialog.OnDateSetListener, TimePickerDialog.OnTimeSetListener {

private var format = DecimalFormat("00")

private var day = 0

private var month = 0

private var year = 0

private var hour12 = 0

private var hour24 = 0

private var minute = 0

private var amPm = ""

private var saveDay = 0

private var saveMonth = 0

private var saveYear = 0

private var saveHour12 = 0

private var saveHour24 = 0

private var saveMinute = 0

private var saveAmPm = ""

init {

dateTime()

}

private fun getDateTimeCalender() {

val calendar: Calendar = Calendar.getInstance()

day = calendar.get(Calendar.DAY\_OF\_MONTH)

month = calendar.get(Calendar.MONTH)+1

year = calendar.get(Calendar.YEAR)

hour24 = calendar.get(Calendar.HOUR\_OF\_DAY)

hour12 = calendar.get(Calendar.HOUR\_OF\_DAY)

amPm = if (hour12 >= 12){

"PM"

}else{

"AM"

}

hour12 %= 12

if (hour12 == 0){

hour12 = 12

}

minute = calendar.get(Calendar.MINUTE)

val dateTime: TextView = noteView.findViewById(R.id.date\_time)

dateTime.text = "${format.format(day)}/${format.format(month)}/${format.format(year)} , ${format.format(hour12)}:${format.format(minute)} $amPm"

println("Curr Date : ${format.format(day)}/${format.format(month)}/${format.format(year)} , ${format.format(hour12)}:${format.format(minute)} $amPm")

month -= 1

}

private fun dateTime() {

getDateTimeCalender()

val dateTime:TextView = noteView.findViewById(R.id.date\_time)

dateTime.setOnClickListener {

val dp = DatePickerDialog(noteView, this, year, month, day)

dp.datePicker.minDate = System.currentTimeMillis()

dp.show()

}

val date: ImageView = noteView.findViewById(R.id.select\_date\_time)

date.setOnClickListener {

val dp = DatePickerDialog(noteView, this, year, month, day)

dp.datePicker.minDate = System.currentTimeMillis()

dp.show()

}

}

override fun onDateSet(view: DatePicker?, year: Int, month: Int, day: Int) {

saveDay = day

saveMonth = month+1

saveYear = year

getDateTimeCalender()

TimePickerDialog(view?.context, this, hour24, minute, false).show()

}

override fun onTimeSet(view: TimePicker?, hour: Int, minute: Int) {

saveAmPm = if (hour >= 12){

"PM"

}else{

"AM"

}

saveHour24 = hour

saveHour12 = hour%12

if (saveHour12 == 0){

saveHour12 = 12

}

saveMinute = minute

val dateTime: TextView = noteView.findViewById(R.id.date\_time)

dateTime.text = "${format.format(saveDay)}/${format.format(saveMonth)}/${format.format(saveYear)} , ${format.format(saveHour12)}:${format.format(saveMinute)} $saveAmPm"

println("Saved Date : ${format.format(saveDay)}/${format.format(saveMonth)}/${format.format(saveYear)} , ${format.format(saveHour12)}:${format.format(saveMinute)} $saveAmPm")

updateDateTime()

}

private fun updateDateTime(){

day = saveDay

month = saveMonth -1

year = saveYear

hour12 = saveHour12

hour24 = saveHour24

minute = saveMinute

amPm = saveAmPm

}

fun getDateTime(): String {

return "${format.format(day)}/${format.format(month+1)}/${format.format(year)} , ${format.format(hour24)}:${format.format(minute)}"

}

}

import android.app.PendingIntent

import android.content.BroadcastReceiver

import android.content.Context

import android.content.Intent

import androidx.core.app.NotificationCompat

import androidx.core.app.NotificationManagerCompat

import com.singhvikrant.mobilecomputingtasks.R

import com.singhvikrant.mobilecomputingtasks.roomDatabase.NoteEntity

import com.singhvikrant.mobilecomputingtasks.ui.MainActivity

class NoteNotification: BroadcastReceiver(){

private val channelId = "remindMe"

private val notificationId = 256

override fun onReceive(context: Context, intent: Intent) {

val bundle = intent.getBundleExtra("bundle")

val title = intent.getStringExtra("title").toString()

var note = intent.getStringExtra("note").toString()

var dateTime = intent.getStringExtra("dateTime").toString()

println("title @ $title")

println("note @ $note")

if (note.isEmpty()){

note = "No description"

}

val noteEntity: NoteEntity = bundle?.getSerializable("noteEntity") as NoteEntity

println("noteEntity $noteEntity")

val newIntent = Intent(context, MainActivity::class.java)

newIntent.putExtra("title", title)

newIntent.putExtra("note", note)

newIntent.putExtra("dateTime", dateTime)

newIntent.putExtra("noteEntity", bundle.getSerializable("noteEntity") as NoteEntity)

val pendingIntent = PendingIntent.getActivity(context, 0, newIntent, PendingIntent.FLAG\_CANCEL\_CURRENT) println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ Working @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@")

val builder = NotificationCompat.Builder(context, channelId) .setSmallIcon(R.mipmap.ic\_launcher\_round)

.setContentTitle(title)

.setContentText(note)

.setAutoCancel(true)

.setContentIntent(pendingIntent) .setPriority(NotificationCompat.PRIORITY\_DEFAULT)

val notificationManager:NotificationManagerCompat = NotificationManagerCompat.from(context)

notificationManager.notify(notificationId, builder.build())

}}

import android.app.\*

import android.content.Intent

import android.os.Build

import android.os.Bundle

import android.view.Gravity

import android.widget.\*

import androidx.appcompat.app.AppCompatActivity

import androidx.lifecycle.ViewModelProvider

import com.google.android.material.textfield.TextInputEditText

import com.google.android.material.textfield.TextInputLayout

import com.singhvikrant.mobilecomputingtasks.R

import com.singhvikrant.mobilecomputingtasks.dateTime.DateTime

import com.singhvikrant.mobilecomputingtasks.notification.NoteNotification

import com.singhvikrant.mobilecomputingtasks.notification.SetAlarm

import com.singhvikrant.mobilecomputingtasks.roomDatabase.NoteEntity

import com.singhvikrant.mobilecomputingtasks.roomDatabase.NoteViewModel

import java.text.SimpleDateFormat

import java.util.\*

class NoteActivity : AppCompatActivity() {

private lateinit var viewModel: NoteViewModel

private val channelId = "remindMe"

private val notificationId = 256

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_note)

createNotificationChannel(channelId, notificationId)

val inputDescription:TextInputEditText = findViewById(R.id.input\_description)

inputDescription.gravity = Gravity.TOP

inputDescription.setHorizontallyScrolling(false)

inputDescription.maxLines = 17

val dt = DateTime(this)

val alarmManager:AlarmManager = getSystemService(ALARM\_SERVICE) as AlarmManager

val saveNote: Button = findViewById(R.id.saveNote)

saveNote.setOnClickListener {

if (validateTitle()) {

val title: TextInputLayout = findViewById(R.id.text\_input\_title)

val titleInput = title.editText?.text.toString()

val note: TextInputLayout = findViewById(R.id.text\_input\_description)

val noteInput = note.editText?.text.toString()

val dateTimeText: TextView = findViewById(R.id.date\_time)

val dateTimeTextInput = dateTimeText.text.toString()

val dateTime = dt.getDateTime()

println(dateTime)

val formatter = SimpleDateFormat("dd/MM/yyyy , HH:mm", Locale.ENGLISH)

val dateTimeToMs :Long= formatter.parse(dateTime)!!.time

println(dateTimeToMs)

val timeOfNoteSaved: Long = System.currentTimeMillis()

println(timeOfNoteSaved)

val delay: Long = 0

println(delay)

val noteEntity = NoteEntity(titleInput, noteInput, dateTimeTextInput)

viewModel = ViewModelProvider(this, ViewModelProvider.AndroidViewModelFactory.getInstance(application))[NoteViewModel::class.java]

viewModel.insertNote(noteEntity)

val intent = Intent(this, NoteNotification::class.java)

val bundle = Bundle()

bundle.putSerializable("noteEntity", noteEntity)

intent.putExtra("title", titleInput)

intent.putExtra("note", noteInput)

intent.putExtra("dateTime", dateTimeTextInput)

intent.putExtra("bundle",bundle)

println("noteEntity $noteEntity")

val pendingIntent = PendingIntent.getBroadcast(this, dateTimeToMs.toInt(), intent,PendingIntent.FLAG\_CANCEL\_CURRENT)

SetAlarm(dateTimeToMs, pendingIntent, alarmManager)

Toast.makeText(this, "Saved successfully", Toast.LENGTH\_SHORT).show()

finish()

}

}

}

fun createNotificationChannel(channelId: String, notificationId: Int) {

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {

// Create the NotificationChannel

val name = channelId

val descriptionText = "This is a reminder notification channel"

val importance = NotificationManager.IMPORTANCE\_DEFAULT

val mChannel = NotificationChannel(channelId, name, importance)

mChannel.description = descriptionText

// Register the channel with the system; you can't change the importance

// or other notification behaviors after this

val notificationManager = getSystemService(NOTIFICATION\_SERVICE) as NotificationManager

// val notificationManager = ContextCompat.getSystemService(NOTIFICATION\_SERVICE) as NotificationManager

notificationManager.createNotificationChannel(mChannel)

} }

private fun validateTitle(): Boolean {

val title:TextInputLayout = findViewById(R.id.text\_input\_title)

val titleInput = title.editText?.text.toString()

return if (titleInput.isEmpty()) {

title.error = "Field can't be empty"

false

} else {

title.error = null

true

}}}

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.widget.ImageView

import android.widget.TextView

import androidx.lifecycle.ViewModelProvider

import com.singhvikrant.mobilecomputingtasks.R

import com.singhvikrant.mobilecomputingtasks.roomDatabase.NoteEntity

import com.singhvikrant.mobilecomputingtasks.roomDatabase.NoteViewModel

class NoteDescriptionActivity : AppCompatActivity() {

private lateinit var viewModel: NoteViewModel

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_note\_description)

val title = intent.getStringExtra("title").toString()

var note = intent.getStringExtra("note").toString()

var dateTime = intent.getStringExtra("dateTime").toString()

val noteEntity: NoteEntity = intent.getSerializableExtra("noteEntity") as NoteEntity

if (note.isEmpty()){

note = "No description"

}

val deleteNote:ImageView = findViewById(R.id.deleteNote)

val titleText:TextView = findViewById(R.id.title)

titleText.text = title

val noteText:TextView = findViewById(R.id.note)

noteText.text = note

val dateTimeText:TextView = findViewById(R.id.dateTime)

dateTimeText.text = dateTime

deleteNote.setOnClickListener {

viewModel = ViewModelProvider(this, ViewModelProvider.AndroidViewModelFactory.getInstance(application))[NoteViewModel::class.java]

viewModel.deleteNote(noteEntity)

finish()

}}}

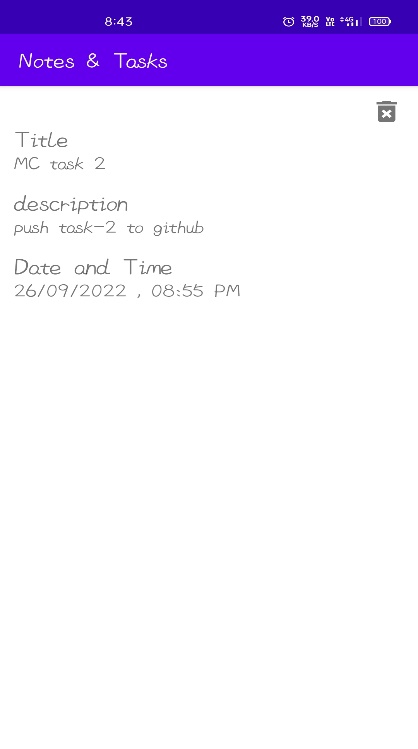
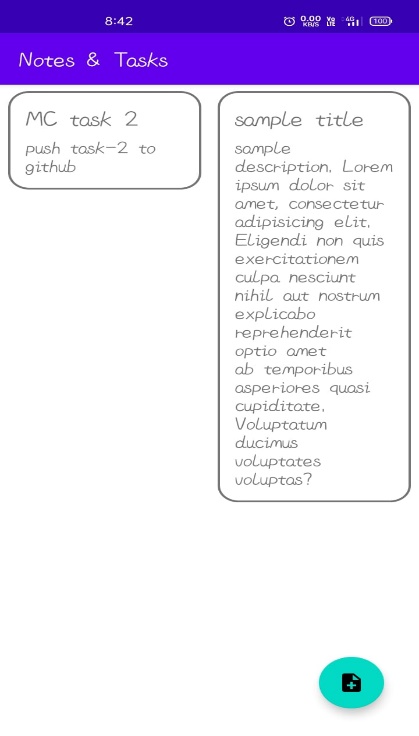
import androidx.room.ColumnInfo

import androidx.room.Entity

* **Output:**

**Graphical user interface

Description automatically generated with low confidence**

****

**Conclusion:**

Hence we successfully created Tab/Features/Facilities/Functionality in our Page as per application and described their working in Laymen’s terms.