

UNIVERSITY of
STIRLING



CSCU9YH
ASSIGNMENT REPORT
NOTE TAKING APP

[3141979]

TABLE OF CONTENTS

1. Introduction	2
2. Structure Of The Application	2
2.1 Fragments (User Interface)	2 - 3
2.2 Main Activity	3
2.3 ViewModel	3
2.4 Database	4
2.5 Page Adapter	4
3. System Overview	5
4. Key Features	5 - 6
5. Implemented Solution	6 - 9
6. References	9
7. Appendix	10 - 27

1 INTRODUCTION

This report discusses the functionality and the implementation of note taking app created using Kotlin in Android Studio. The application uses SQLite for persistent data storage, ensuring that the users notes are preserved even after the app is closed.

Objectives

The main objective of this project was to create a functional and user-friendly note-taking application that allows users to select a date, add notes for that date, and view previously added notes.

The application has met all of these objectives and has additional functionalities that helps to enhance the user's experience.

2 STRUCTURE OF THE APPLICATION

The application is structured with the components such as fragments which handles user interactions and displays the application (UI), the ViewModel which manages the UI and the database, and the database which maintains the storage of notes.

2.1 Fragments (User Interface)

These are responsible for displaying the user interface:

1) FirstFragment: The fragments main functionality is to have a date picker for selecting a date and has a button that helps to navigate to the second fragment to create notes when required.

2) SecondFragment: The fragments functionality includes creating and editing notes. To implement its functionality the fragment has:

- A TextView that displays the selected date from the first fragment
- Two EditTexts for entering the title and content of a note.
- Buttons for saving the note and clearing the input texts.

3) ThirdFragment: The fragment is mainly concerned with the listing of the notes and handling operations related to them. To implement its functionality the fragment has:

- A ListView that displays the list of notes that are arranged according to their date. The list has the date and title of the notes, the user can view the notes by tapping the note from the list and after which the user will be taken to the second fragment where they can edit the notes if required.

- Additionally, this fragment includes a SearchView for searching notes according to the notes title or the content.
- There is a button that allows the user to create a new note which on clicking will direct them to the first fragment to choose a date.
- The user can delete any selected note through a long press if needed

Navigation for Creating a Note

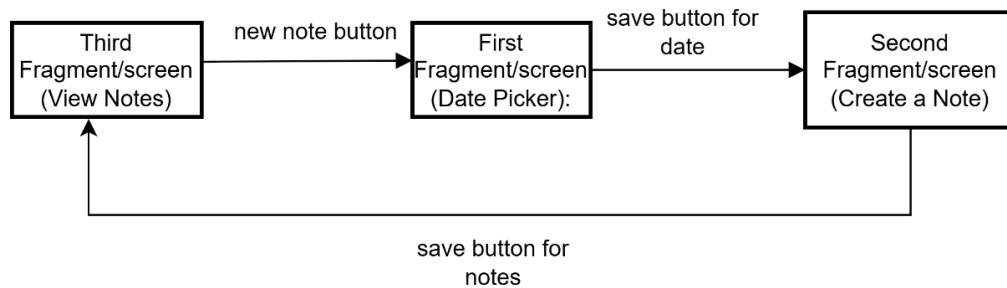


Figure 1. The navigation flow between the fragments for creating a note in the app

As shown in the figure, the user interface has buttons that helps with intuitive navigation for the user.

2.2 Main Activity

The main activity is the entry point for the app and manages the fragments and navigation of the application. It sets up the toolbar, tab layout, and view pager with an adapter (PagerAdapter). It also defines a function navigateFragments to allow the navigation between the different fragments.

2.3 ViewModel

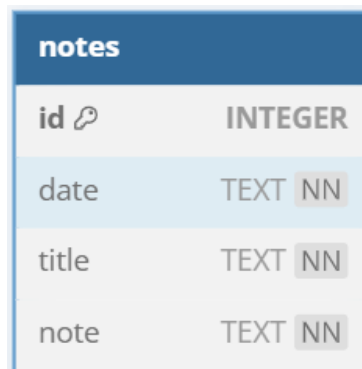
MyViewModel manages the state of the app and works between the UI (fragments) and the database layer. It interacts with the DatabaseAdapter to handle all database operations, including getting, adding, updating, and deleting notes.

The ViewModel provides data to the fragments and uses LiveData objects with selected date, notes, and the current note id, to hold the list of notes and notify the UI when notes are added, updated, or deleted.

2.4 Database

The database handles the operations to store and retrieve data using SQLite:

- DatabaseHelper: This class manages the creation, update and deletion of the SQLite database. It defines the schema for the notes table such as the columns for id, date, and note (figure 2).

A diagram of a SQLite table named 'notes'. The table has a dark blue header with the name 'notes' in white. Below the header, the table structure is shown with columns and their data types. The 'id' column is the primary key, indicated by a key icon, and is of type 'INTEGER'. The 'date', 'title', and 'note' columns are of type 'TEXT' and are marked as 'NN' (Not Null).


notes	
id 	INTEGER
date	TEXT NN
title	TEXT NN
note	TEXT NN

Figure 2. The structure of the notes table, which stores information about the notes.

- DatabaseAdapter: Provides methods for interacting with the database, such as:
 - insertNote: Inserts a new note into the database.
 - updateNote: Updates a previously added note.
 - deleteNote: Deletes a note from the database.
 - getAllNotes: Retrieves all notes from the database.

2.5 Page Adapter

The PageAdapter handles the navigation between different screens in the application using the ViewPager2. It displays the fragments based on the current position, with a position for each screen in the application.

3. SYSTEM OVERVIEW

The application is designed to offer users a way to organize and manage their notes effectively. The system has the following core components (as shown in figure 3) to achieve this:

- Model: DatabaseHelper class is the model and handles the interactions with the SQLite database for storing and receiving the notes.
- View: The view is the three fragments FirstFragment, SecondFragment, and ThirdFragment.
- ViewModel: The MyViewModel class manages and communicates between the model and the view.

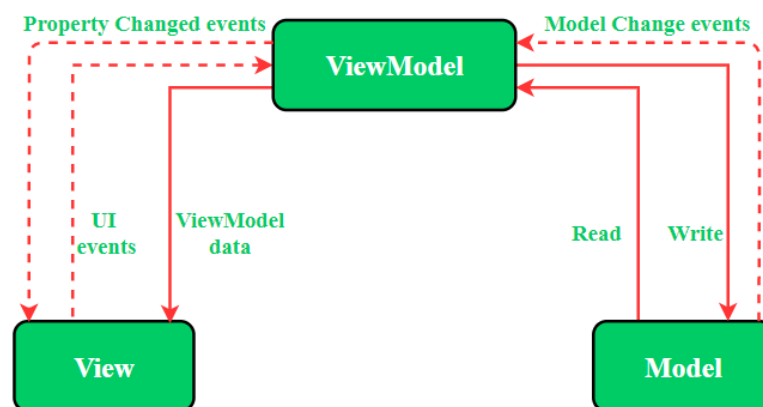


Figure 3. Model View ViewModel Architecture [1]

4. KEY FEATURES

This application has implemented all the basic features required as listed below and has additional features that can enhance the user experience:

Basic

1. **Date Selection:** Users can select a date using a Date Picker in the first fragment. Selected dates are passed to the other fragments through the ViewModel.
2. **Note Taking:** Users can add notes with the selected date in the second fragment. Notes are saved with their selected date and stored accordingly.
3. **Note Listing:** The third fragment displays a list of notes categorized by date.

Advanced

4. **Persistent Data:** The data will remain stored on the device and will retain it even after the device shuts down. A database is used for this with SQLite.
5. **Search Function:** Users can search for notes by their title or the content with the help of SearchView.
6. **Editing Notes:** Users can edit existing notes by selecting them from the list (onclick) and they will be directed to the Add Note screen with content of the note.
7. **Deleting Notes:** Users can delete the notes they don't require (long press). A confirmation message will be given asking the user to cancel the action or to delete, to ensure that the note will not be deleted accidentally.

I have also added a title in the Add Notes screen because if the full content of each note is displayed, it could be lengthy and make the interface not user friendly. The notes list hence has both titles and the note content, but only the titles are displayed in third fragment.

Users can click on the title to view the complete content in the add notes screen, making the interface both efficient and have a clean and organized list that is easy to navigate.

5. IMPLEMENTED SOLUTION

Justification of design choices and better alternatives

Design choice of using SQLite database was preferred over shared preferences and other data structures due to its nature of persistent data.

The user interface is simplistic in nature making it more intuitive for the user, and the navigation implemented using buttons increases the performance.

In the implemented solution there are still some better alternatives, over the ones used such as:

- 1) Room Persistence Library provides an abstraction layer on the SQLite to allow database access while being able to use the full advantages of SQLite [2]. It has annotations that can help reduce the redundant code and the queries are also verified at compile time.
- 2) RecyclerView can be used for a better way to display list of notes in the third fragment as the layout could be more user-friendly.

Assumptions

Some of the assumptions made are:

- Only one note can be selected for deleting at a time.
- Users will use the app on smartphones.
- Users will not go past a certain note length limit.

Testing

The application was tested for each of these features after the implementation:

1) For its functionality: Creating a note, editing, viewing, searching, and deletion of notes.

The application was able to perform well regarding its functionality.

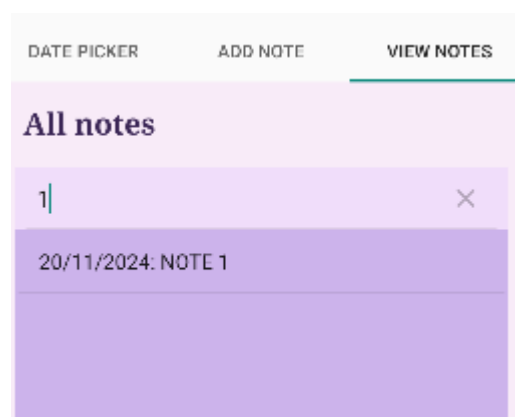
2) Usability Testing: If the user interface is intuitive, if appropriate messages are given when faced with an error?

The application was able to perform well on the medium smartphone in android studio but faced difficulty in the other devices, and appropriate error messages are given.

3) Edge Case Testing: Testing with empty notes, long notes, invalid input.

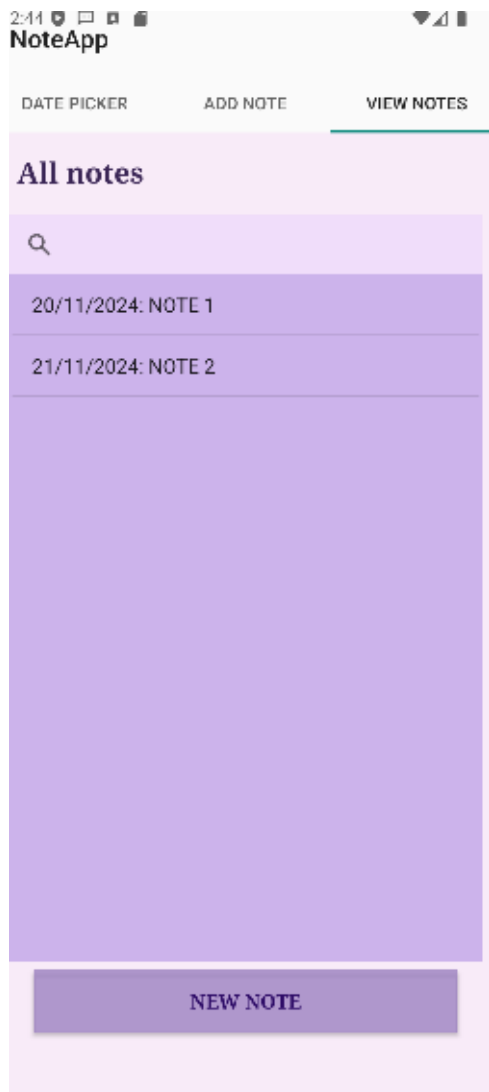
On testing the application, it was found that the empty notes can be saved as long as it has a title. Longer notes can be added, and the user can scroll to view them. The note cannot be saved without a title so the notes with invalid inputs will not be saved.

Screenshots of the application

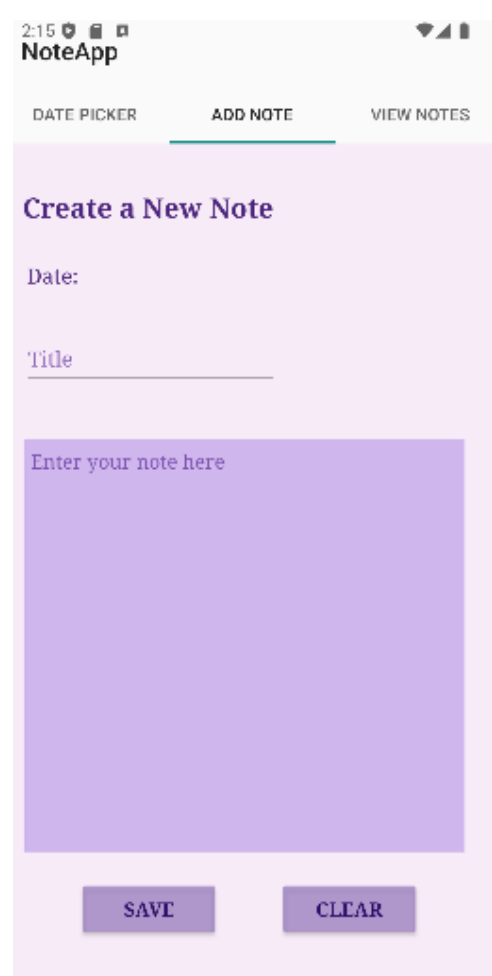


The search functionality of the application in the View Notes screen

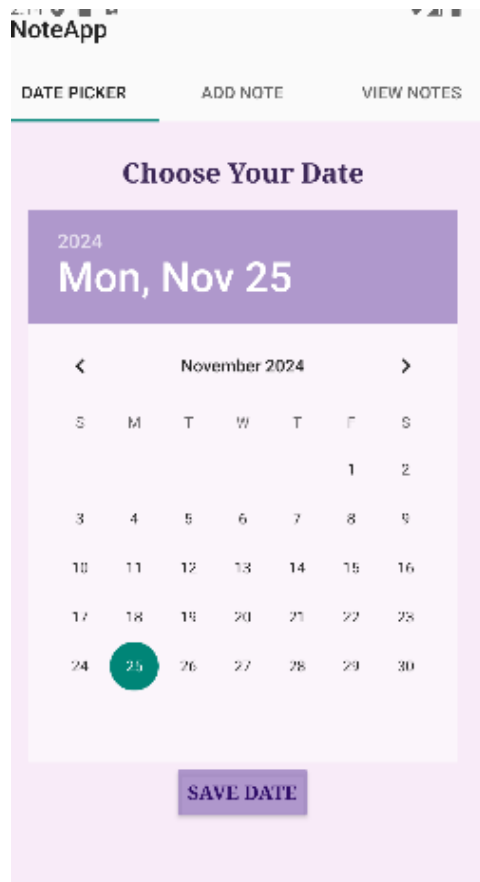
The View Notes Screen



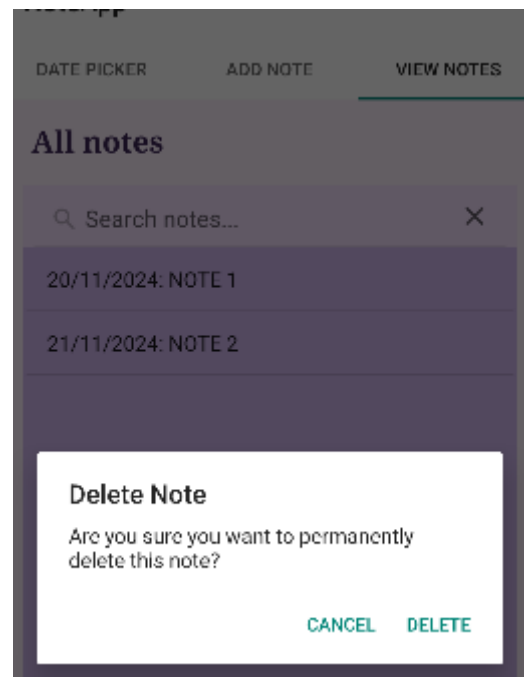
The Add Note Screen



The Date Picker Screen



The delete note functionality in the View Notes screen



6. REFERENCES

[1] R. Mishra, "MVVM (Model View ViewModel) Architecture Pattern in Android," *GeeksforGeeks*, Oct. 29, 2020.

<https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/>

[2] Google, "Save data in a local database using Room | Android Developers," *Android Developers*, 2019.

<https://developer.android.com/training/data-storage/room>

7. APPENDIX

DatabaseAdapter:

```
package com.example.noteApp

import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import java.sql.SQLException

class DatabaseAdapter(private val context: Context) {

    private lateinit var dbHelper: DatabaseHelper
    private lateinit var db: SQLiteDatabase

    @Throws(SQLException::class)
    fun open(): DatabaseAdapter { //Opens a connection to the database
        dbHelper = DatabaseHelper(context)
        db = dbHelper.writableDatabase
        return this
    }

    fun close() {
        dbHelper.close() //Closes the database connection
    }

    //deletes a note from the database based on the note's id
    fun deleteNote(id: Long): Boolean {
        val where = "${DatabaseHelper.COLUMN_ID}=$id"
        return db.delete(DatabaseHelper.TABLE_NOTES, where, null) > 0
    }

    //Adds a new note into the database with the all the note details
    fun insertNote(date: String, title: String, note: String): Long {
        val values = ContentValues().apply {
            put(DatabaseHelper.COLUMN_DATE, date)
            put(DatabaseHelper.COLUMN_TITLE, title)
        }
    }
}
```

```

        put(DatabaseHelper.COLUMN_NOTE, note)
    }
    return db.insert(DatabaseHelper.TABLE_NOTES, null, values)
}

// Updates previously added note in the database
fun updateNote(id: Long, date: String, title: String, note: String): Boolean {
    val values = ContentValues().apply {
        put(DatabaseHelper.COLUMN_DATE, date)
        put(DatabaseHelper.COLUMN_TITLE, title)
        put(DatabaseHelper.COLUMN_NOTE, note)
    }
    val where = "${DatabaseHelper.COLUMN_ID}=$id"
    return db.update(DatabaseHelper.TABLE_NOTES, values, where, null) > 0
}

//Gets all notes from the database
fun getAllNotes(): Map<String, List<Triple<Long, String, String>>> {
    val notes =
        mutableMapOf<String, MutableList<Triple<Long, String, String>>>() // Create a
mutable map to group notes according to their date
    val columns = arrayOf(
        DatabaseHelper.COLUMN_ID,
        DatabaseHelper.COLUMN_DATE,
        DatabaseHelper.COLUMN_TITLE,
        DatabaseHelper.COLUMN_NOTE
    )

    val cursor = db.query( // Query the database table for all notes, date in ascending order
        DatabaseHelper.TABLE_NOTES,
        columns,
        null,
        null,

```

```

        null,
        null,
        "${DatabaseHelper.COLUMN_DATE} ASC"
    )

    cursor.use {
        while (cursor.moveToNext()) {
            val id = cursor.getLong(0)
            val date = cursor.getString(1)
            val title = cursor.getString(2)
            val note = cursor.getString(3)

            if (!notes.containsKey(date)) {
                notes[date] = mutableListOf()
            }
            notes[date]?.add(Triple(id, title, note))
        }
    }

    return notes
}
}

```

DatabaseHelper:

```

package com.example.noteApp

import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, "notes_db", null, 1) {
    companion object {
        const val TABLE_NOTES = "notes" //Table name
        const val COLUMN_ID = "id" // note id
        const val COLUMN_DATE = "date" //date for the note
        const val COLUMN_TITLE = "title" // title of note
    }
}

```

```

        const val COLUMN_NOTE = "note" // content of the note
    }

    //Creates the database table if it does not exist before
    override fun onCreate(db: SQLiteDatabase) {
        val createTable = """
            CREATE TABLE $TABLE_NOTES (
                $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $COLUMN_DATE TEXT NOT NULL,
                $COLUMN_TITLE TEXT NOT NULL,
                $COLUMN_NOTE TEXT NOT NULL
            );
        """.trimIndent()
        db.execSQL(createTable)
    }

    //Upgrades the database according to the version
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS $TABLE_NOTES")
        onCreate(db)
    }
}

```

FirstFragment:

```

package com.example.noteApp

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.DatePicker
import androidx.lifecycle.ViewModelProvider
class FirstFragment : Fragment() {
    private lateinit var viewModel: MyViewModel

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        val view = inflater.inflate(R.layout.fragment_first, container, false)
        viewModel = activity?.run { ViewModelProvider(this)[MyViewModel::class.java]
        } ?: throw Exception("Invalid Activity")

        // Initialising views
        val datePicker = view.findViewById<DatePicker>(R.id.datePicker)
        val saveButton = view.findViewById<Button>(R.id.saveButton)
    }
}

```

```

        saveButton.setOnClickListener {
            val Date = "${datePicker.dayOfMonth}/${datePicker.month +
1}/${datePicker.year}"// Getting the date from the date picker
            viewModel.setDate(Date)
            (activity as? MainActivity)?.navigateFragments(1) // navigating to second fragment to
add a note with the date
        }

        return view
    }
}

```

SecondFragment:

```

package com.example.noteApp

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.AdapterView
import android.widget.AdapterView.OnItemClickListener
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.lifecycle.ViewModelProvider
class SecondFragment : Fragment() {
    private lateinit var viewModel: MyViewModel

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        val view = inflater.inflate(R.layout.fragment_second, container, false)
        viewModel = activity?.run { ViewModelProvider(this)[MyViewModel::class.java]
        } ?: throw Exception("Invalid Activity")

        // Initialising views
        val dateTextView = view.findViewById<TextView>(R.id.dateTextView)
        val titleEditText = view.findViewById<EditText>(R.id.titleEditText)
        val noteEditText = view.findViewById<EditText>(R.id.noteEditText)
        val saveButton = view.findViewById<Button>(R.id.saveButton)
        val clearButton = view.findViewById<Button>(R.id.clearButton)

        viewModel.selectedDate.observe(viewLifecycleOwner) { date ->
            dateTextView.text = if (date.isNotEmpty()) "Date: $date" else "Select Date"
        }
    }
}

```

```

    }

    viewModel.selectedId.observe(viewLifecycleOwner) { noteId ->
        if (noteId != null) {
            val date = viewModel.selectedDate.value // Get the date from ViewModel
            date?.let { selectedDate ->
                val notes = viewModel.notes.value // Get the note from ViewModel
                // Find a note based on the selected date and its id
                val noteTriple = notes?.get(selectedDate)?.find { it.first == noteId }
                titleEditText.setText(noteTriple?.second.orEmpty())
                noteEditText.setText(noteTriple?.third.orEmpty())
            }
        } else {
            titleEditText.text.clear()
            noteEditText.text.clear()
        }
    }

    saveButton.setOnClickListener { //to save a note
        val date = viewModel.selectedDate.value
        val title = titleEditText.text.toString().trim()
        val note = noteEditText.text.toString().trim()
        if (date != null && date.isNotEmpty() && title.isNotEmpty()) {
            viewModel.addAndUpdate(date, title, note) // Call the ViewModel method to add/
            update the note
            Toast.makeText(requireContext(), "Note saved", Toast.LENGTH_SHORT).show()
            //message to show success
            (activity as? MainActivity)?.navigateFragments(2) //navigates to 3rd fragment to
            see the note in the list
        } else {
            Toast.makeText(requireContext(), "Date or title is missing",
            Toast.LENGTH_SHORT).show() //message when info is missing
        }
    }

    clearButton.setOnClickListener { //clear button clear any inputs
        noteEditText.text.clear()
    }

    return view
}
}

```

ThirdFragment:

```
package com.example.noteApp
```



```

import android.app.AlertDialog
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.ListView
import android.widget.SearchView
import android.widget.Toast
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProvider

class ThirdFragment : Fragment() {
    private lateinit var viewModel: MyViewModel
    private lateinit var adapter: ArrayAdapter<String>
    private val notesList = mutableListOf<String>()
    private val filteredList = mutableListOf<String>()
    private val noteIds = mutableMapOf<Int, Long>()
    private val Content = mutableMapOf<Int, String>()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        val view = inflater.inflate(R.layout.fragment_third, container, false)

        // Initialising views
        val notesListView = view.findViewById<ListView>(R.id.notesListView)
        val searchView = view.findViewById<SearchView>(R.id.searchView)
        val NoteButton = view.findViewById<Button>(R.id.noteButton)
        adapter = ArrayAdapter(requireContext(), android.R.layout.simple_list_item_1,
filteredList)
        notesListView.adapter = adapter

        viewModel = activity?.run {
            ViewModelProvider(this)[MyViewModel::class.java]
        } ?: throw Exception("Invalid Activity")

        viewModel.notes.observe(viewLifecycleOwner) { notes ->
            updateNotes(notes)
        }

        // The note button takes the user to the date picker fragment to select a date
        NoteButton.setOnClickListener {
            viewModel.setDate("")
            viewModel.setNoteId(null)

```

```

        (activity as? MainActivity)?.navigateFragments(0)
    }

    // The search functionality listener
    searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {
        override fun onQueryTextSubmit(query: String?) = false
        override fun onQueryTextChange(newText: String?): Boolean {
            filterNotes(newText)
            return true
        }
    })

    notesListView.setOnItemClickListener { _, _, position, _ ->
        val noteId = noteIds[position] ?: return@setOnItemClickListener
        selectNote(position, noteId)
    }

    // To delete notes, user has to long press the note
    notesListView.setOnItemLongClickListener { _, _, position, _ ->
        DeleteNote(position)
        true
    }

    return view
}

//Updates the notes list and maps with info from the ViewModel
private fun updateNotes(notes: Map<String, List<Triple<Long, String, String>>>) {
    notesList.clear()
    filteredList.clear()
    noteIds.clear()
    Content.clear()
    var position = 0

    notes.forEach { (date, noteList) ->
        noteList.forEach { (id, title, note) ->
            val displayText = "$date: $title"
            notesList.add(displayText)
            noteIds[position] = id
            Content[position] = "$title $note"
            position++
        }
    }
    filteredList.addAll(notesList)
    adapter.notifyDataSetChanged() // Update adapter for the new data
}

// filter the notes based on the search information

```

```

private fun filterNotes(query: String?) {
    filteredList.clear()
    if (query.isNullOrEmpty()) {
        filteredList.addAll(notesList)
        notesList.indices.forEach { index -> noteIds[index] = noteIds[index] ?: -1 }
    } else {
        val searchQuery = query.lowercase()
        notesList.forEachIndexed { index, displayText -> // Add notes to the filtered list if
they match the search
            if (Content[index]?.lowercase()?.contains(searchQuery) == true) {
                filteredList.add(displayText)
                noteIds[filteredList.lastIndex] = noteIds[index] ?: -1
            }
        }
    }
    adapter.notifyDataSetChanged()
}

// Method to select notes for viewing or editing
private fun selectNote(position: Int, noteId: Long) {
    val selectedNote = filteredList[position]
    val date = selectedNote.split(":")[0].trim()

    viewModel.setDate(date)
    viewModel.setNoteId(noteId)
    (activity as? MainActivity)?.navigateFragments(1)
}

// Method to delete note after the user confirms
private fun DeleteNote(position: Int) {
    val noteId = noteIds[position] ?: return

    AlertDialog.Builder(requireContext()) //dialog
        .setTitle("Delete Note")
        .setMessage("Are you sure you want to permanently delete this note?")
        .setPositiveButton("Delete") { _, _ ->
            viewModel.deleteNote(noteId) // Deleting the note using ViewModel
            Toast.makeText(requireContext(), "Note deleted", Toast.LENGTH_SHORT).show()
        }
        .setNegativeButton("Cancel", null)
        .show()
}

override fun onResume() { // Refreshes the adapter
    super.onResume()
    adapter.notifyDataSetChanged()
}
}

```

MainActivity:

```
package com.example.noteApp

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.widget.Toolbar
import androidx.viewpager2.widget.ViewPager2
import com.google.android.material.tabs.TabLayout
import com.google.android.material.tabs.TabLayoutMediator
class MainActivity : AppCompatActivity() {
    private lateinit var viewPager: ViewPager2
    private lateinit var adapter: PageAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        // Toolbar for the app
        val toolbar = findViewById<Toolbar>(R.id.toolbar)
        setSupportActionBar(toolbar)

        // Creating tabLayout and giving the tabs names according to functionality
        val tabLayout = findViewById<TabLayout>(R.id.tab_layout)
        tabLayout.addTab(tabLayout.newTab().setText("Date Picker"))
        tabLayout.addTab(tabLayout.newTab().setText("Add Note"))
        tabLayout.addTab(tabLayout.newTab().setText("View Notes"))
        tabLayout.tabGravity = TabLayout.GRAVITY_FILL

        viewPager = findViewById(R.id.pager)
        adapter = PageAdapter(this, 3) // a tab for each fragment
        viewPager.adapter = adapter

        TabLayoutMediator(tabLayout, viewPager) { tab, position ->
            tab.text = when (position) {
                0 -> "Date Picker"
                1 -> "Add Note"
                2 -> "View Notes"
                else -> "Tab $position"
            }
        }.attach()

        viewPager.currentItem = 2 // The app will start at the third fragment so that user can
        create and see existing notes

    }
    override fun onDestroy() {
        super.onDestroy()
    }
}
```

```

    }
    //function that allows the navigation to a specific fragment
    fun navigateFragments(index: Int) {
        viewPager.currentItem = index
    }
}

```

MyViewModel:

```

package com.example.noteApp

import android.app.Application
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData

class MyViewModel(application: Application) : AndroidViewModel(application) {
    private val dbAdapter: DatabaseAdapter = DatabaseAdapter(application).apply { open() }

    private val Date = MutableLiveData<String>() // stores the date
    val selectedDate: LiveData<String> = Date

    //stores list of notes according to date
    private val note = MutableLiveData<Map<String, List<Triple<Long, String, String>>>>()
    val notes: LiveData<Map<String, List<Triple<Long, String, String>>>> = note

    private val NoteId = MutableLiveData<Long?>() // stores the id of the note
    val selectedId: LiveData<Long?> = NoteId

    init {
        loadNotes()
    }
    private fun loadNotes() {
        note.value = dbAdapter.getAllNotes() // Loads all notes from the database and updates
        them
    }
    fun setDate(date: String) {
        Date.value = date
    }
    fun setNoteId(id: Long?) {
        NoteId.value = id
    }
    fun addAndUpdate(date: String, title: String, note: String) {
        val noteId = NoteId.value
        if (noteId != null) {
            dbAdapter.updateNote(noteId, date, title, note) // Updates the existing note
        } else {
            dbAdapter.insertNote(date, title, note) // if note id not there then insert a new note
        }
    }
}

```

```

        loadNotes()
        NoteId.value = null
    }

    fun deleteNote(noteId: Long) {
        dbAdapter.deleteNote(noteId) // Deletes note based on its id
        loadNotes()
    }

    override fun onCleared() {
        super.onCleared()
        dbAdapter.close() // Closes the database connection
    }
}

```

PagerAdapter:

```

package com.example.noteApp

import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentActivity
import androidx.viewpager2.adapter.FragmentStateAdapter

class PageAdapter(fa: FragmentActivity, private val mNumOfTabs: Int) :
    FragmentStateAdapter(fa) {
    override fun getItemCount(): Int = mNumOfTabs

    override fun createFragment(position: Int): Fragment {
        return when (position) {
            0 -> FirstFragment() //selecting date
            1 -> SecondFragment() //adding note
            2 -> ThirdFragment() // notes list
            else -> ThirdFragment()
        }
    }
}

```

Layout files:

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

```

```

<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:elevation="6dp"
/>

<com.google.android.material.tabs.TabLayout
    android:id="@+id/tab_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:tabMode="fixed"
    app:tabGravity="fill"
    android:contentDescription="Tab layout" />

<androidx.viewpager2.widget.ViewPager2
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
/>

</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#F8EBF8"
    android:orientation="vertical">

    <TextView
        android:id="@+id/heading"
        android:layout_width="match_parent"
        android:layout_height="72dp"
        android:fontFamily="serif"
        android:text="\nChoose Your Date"
        android:textAlignment="center"
        android:textColor="#DC1D073C"
        android:textSize="22sp"
        android:textStyle="bold" />

    <DatePicker
        android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="449dp"

```

```
android:layout_gravity="center"
android:background="#7BF4EEEE"
android:backgroundTint="#FFFFFF"
android:headerBackground="#D5A086C3" />
```

```
<Button
    android:id="@+id/saveButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:backgroundTint="#D5A086C3"
    android:fontFamily="serif"
    android:text="Save Date"
    android:textColor="#2F0D68"
    android:textSize="16sp"
    android:textStyle="bold" />
```

```
</LinearLayout>
```

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#F8EBF8"
    android:padding="16dp">
```

```
<TextView
    android:id="@+id/dateTextView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="80dp"
    android:fontFamily="serif"
    android:text=" Date: "
    android:textColor="#4D277E"
    android:textSize="17sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<EditText
    android:id="@+id/noteEditText"
    android:layout_width="366dp"
    android:layout_height="344dp"
    android:background="@android:drawable/edit_text"
```



```

android:backgroundTint="#CCB3EB"
android:fontFamily="serif"
android:gravity="top"
android:hint="Enter your note here"
android:minHeight="120dp"
android:padding="8dp"
android:textColorHint="#9C4A148C"
android:textSize="16sp"
app:layout_constraintBottom_toTopOf="@id/buttonContainer"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.0"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/dateTextView"
app:layout_constraintVertical_bias="0.367" />

```

```

<LinearLayout
    android:id="@+id/buttonContainer"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="horizontal"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent">

```

```

</LinearLayout>

```

```

<Button
    android:id="@+id/clearButton"
    android:layout_width="116dp"
    android:layout_height="49dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="52dp"
    android:layout_weight="1"
    android:backgroundTint="#D5A086C3"
    android:fontFamily="serif"
    android:text="Clear"
    android:textColor="#2F0D68"
    android:textSize="16sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/noteEditText"
    app:layout_constraintVertical_bias="0.0"
    tools:text="Clear" />

```

```

<Button
    android:id="@+id/saveButton"

```

```

        android:layout_width="116dp"
        android:layout_height="49dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="48dp"
        android:layout_weight="1"
        android:backgroundTint="#D5A086C3"
        android:fontFamily="serif"
        android:text="Save"
        android:textColor="#2F0D68"
        android:textSize="16sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toStartOf="@+id/clearButton"
        app:layout_constraintTop_toBottomOf="@+id/noteEditText" />

```

<EditText

```

        android:id="@+id/titleEditText"
        android:layout_width="208dp"
        android:layout_height="48dp"
        android:layout_marginStart="2dp"
        android:layout_marginTop="33dp"
        android:layout_marginBottom="3dp"
        android:ems="10"
        android:fontFamily="serif"
        android:hint="Title"
        android:inputType="text"
        android:textColor="#4D277E"
        android:textColorHint="#9C4A148C"
        android:textSize="17sp"
        app:layout_constraintBottom_toTopOf="@+id/noteEditText"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/dateTextView"
        app:layout_constraintVertical_bias="0.0" />

```

<TextView

```

        android:id="@+id/TextView"
        android:layout_width="240dp"
        android:layout_height="31dp"
        android:layout_marginStart="1dp"
        android:layout_marginTop="21dp"
        android:layout_marginEnd="155dp"
        android:layout_marginBottom="28dp"
        android:fontFamily="serif"
        android:text="Create a New Note"
        android:textColor="#4D277E"
        android:textSize="22sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/dateTextView"
        app:layout_constraintEnd_toEndOf="parent"

```

```

        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#F8EBF8"
    android:padding="16dp">

    <TextView
        android:id="@+id/TextView"
        android:layout_width="261dp"
        android:layout_height="wrap_content"
        android:height="50dp"
        android:ems="10"
        android:fontFamily="serif"
        android:includeFontPadding="true"
        android:text=" All notes"
        android:textColor="#DC1D073C"
        android:textSize="23sp"
        android:textStyle="bold" />

    <SearchView
        android:id="@+id/searchView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3BD3AEFF"
        android:queryHint="Search notes..." />

    <ListView
        android:id="@+id/notesListView"
        android:layout_width="wrap_content"
        android:layout_height="552dp"
        android:background="#CCB3EB"
        android:clipToPadding="false"
        android:paddingHorizontal="2sp" />

    <Button
        android:id="@+id/noteButton"
        android:layout_width="349dp"
        android:layout_height="62dp"
        android:layout_gravity="center"
        android:backgroundTint="#D5A086C3"
        android:fontFamily="serif"
        android:text="New Note"
        android:textColor="#2F0D68"
        android:textSize="16sp"
        android:textStyle="bold" />

```

</LinearLayout>