Note-It Mobile Android App

Members
Edison CHAN (300370372)
Nidhi (300378175)

Content Page

Content Page 1

1. Project description chapter (Functionality) 2

2. Project design chapter 3

2.1 Introduction 3

2.2 Features 3

2.3 Implementation 3

3. Implementation (coding) chapter 4

3.1 Main Flow 4

3.2 Activities 5

3.3 Navigation 8

3.3 Date Picker 9

3.4 Calendar 11

3.5 Database Section 13

3.6 Notification Section (Disabled) 15

3.7 Source Control 17

4. Testing chapter 18

4.1 Main Page 18

4.2 Note Add/Edit 19

4.3 Calendar View 20

4.4 To do List by Date View 21

4.5 Update Notes, Note Adapter 22

4.6 Notification Triggers 23

5. User guide chapter 24

5.1 Main Screen (to Do list) 24

5.2 Add Notes 25

5.3 Update Notes 26

5.4 Calendar View 27

5.5 To do List by Date View 28

6. Conclusion section 29

7. Reference 30

1. Project description chapter (Functionality)

What are we making? Note Taking App
What functionality do we have?

The app is designed to help users take notes on their Android devices. The app will allow users to create, edit, and delete notes, as well as organize them into categories. The app will also provide users with the ability to search through their notes and customize the app's user interface.

The app will be built using Java and the Android Studio development environment. The app will use an ArrayList to link to a ListView with the help of an ArrayAdapter. The app will also use an Intent to jump between two Activities and send data through Intents. The app will use a multiline EditText and change its text orientations. The app will also add permanent storage to the app using SharedPreferences. The app will use the AlertDialog library and add the Menu functionality. The app will use onItemLongClickListener() on Views. The app will also use addTextChanged() and newTextWatcher() to check the behavior of text changing.

The source code has been attached as src.zip in the blackboard submission, and also on github https://github.com/WCEdison/CSIS3175NoteitProject.git

Here are the main components required for the app:

- ListView: To display the created notes.
- Menu: To add a note.
- A secondary Activity: To actually add a note.
- EditText: To create/edit a note.

2. Project design chapter

Why are we picking these functions to work with?
Why are they important and useful to the user?
What are the artistic elements for this project? Purple, Unified color, blah blah

2.1 Introduction

The note-taking app is designed to help users take notes on their Android devices. The app will allow users to create, edit, and delete notes, as well as organize them into categories. The app will also provide users with the ability to search through their notes and customize the app's user interface.

2.2 Features

- Create, edit, and delete notes.
- Organize notes into categories.
- Search through notes.
- Customize the app's user interface.
- View notes due on a specific date using the calendar function.

2.3 Implementation

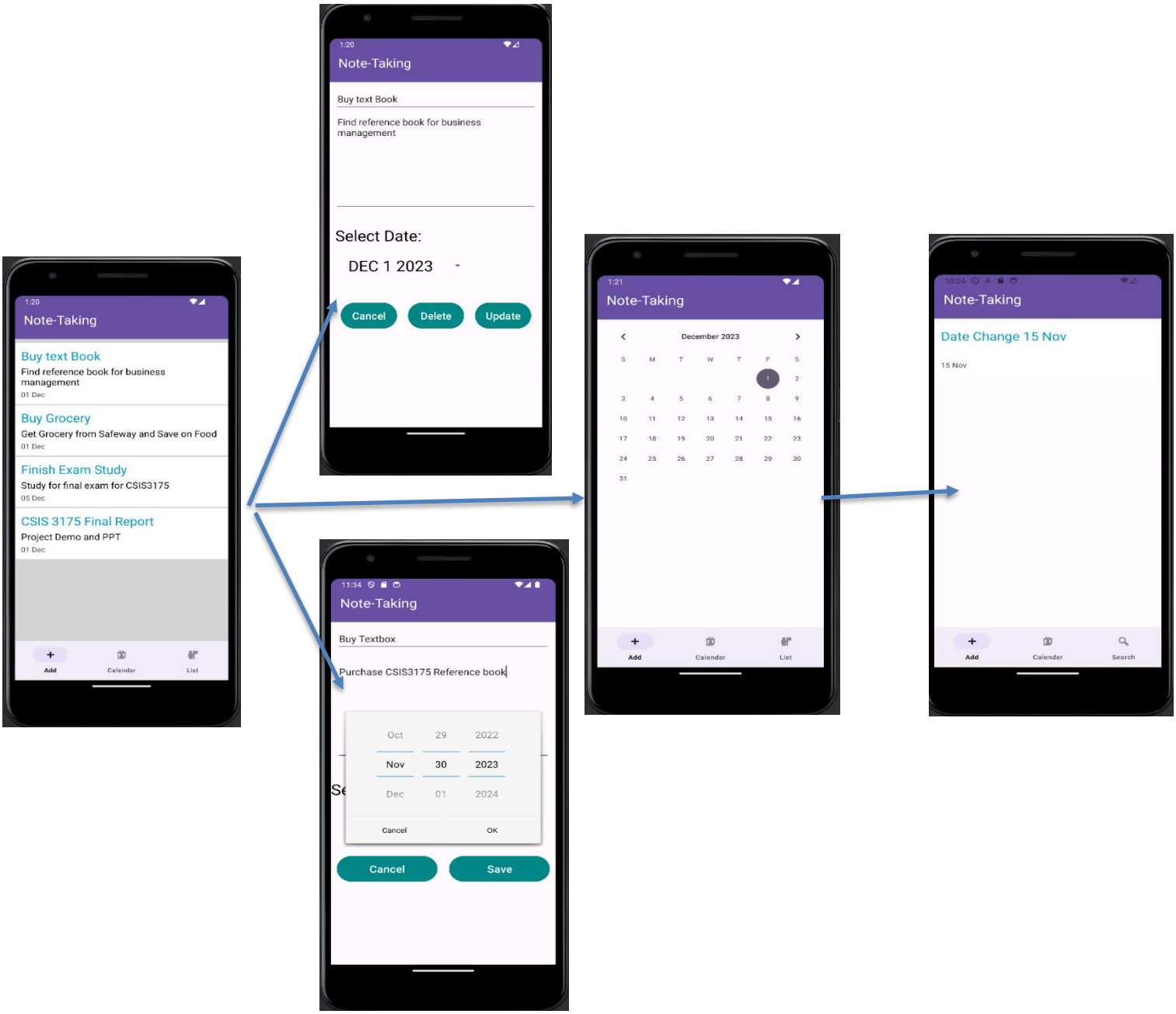
The app will be built using Java and the Android Studio development environment. The app will use an ArrayList to link to a ListView with the help of an ArrayAdapter. The app will also use an Intent to jump between two Activities and send data through Intents. The app will use a multiline EditText and change its text orientations. The app will also add permanent storage to the app using SharedPreferences. The app will use the AlertDialog library and add the Menu functionality. The app will use onItemLongClickListener() on Views. The app will also use addTextChanged() and newTextWatcher() to check the behavior of text changing.

The app will use SQLite database to store notes. The app will use the CalendarView widget to display the calendar. The app will use the DatePickerDialog to select a date. The app will use the CursorAdapter to display the notes due on a specific date.

3. Implementation (coding) chapter

3.1 Main Flow

The system consist of 5 screens, Main Page, Calendar View, Add/Edit View, To do List by Date View and View Note Adapter from Calendar



3.2 Activities

Main Activities

We have done the following in main activity

1. Attach Database
2. Add failsafe for note incorrectly Added
3. Toast messages when list is opened
4. Trigger notification

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  setContentView(R.layout.activity_main);

  db = new NoteDatabaseHelper(this);
  notesListView = findViewById(R.id.notes_list_view);
  ArrayList<Note> notes = db.getAllNotes();
  if (notes.size()==0){
     Random random = new Random();
     int id = random.nextInt(9000) + 1000;
     db.addNote(new Note(id, "title", "description"));
     //finish();
  }
  Toast.makeText(MainActivity.this, "Welcome, you have: " + notes.size() + " events.",
Toast.LENGTH_SHORT).show();
  TriggerNotification("Note it update!", notes.size() + " events.");

  updateUI();
  createNotificationChannel();
  notesListView.setOnItemClickListener((parent, view, position, id) -> {
     Note note = (Note) parent.getItemAtPosition(position);
     Intent intent = new Intent(MainActivity.this, NoteActivity.class);
     intent.putExtra("NOTE_ID", note.getId());
     startActivity(intent);
  });
  // findViewById(R.id.fab).setOnClickListener(view -> startActivity(new Intent(MainActivity.this,
AddNoteActivity.class)));

  setBottomNav();
}
```

We implemented Adapter to Display Notes in different views (Main Page vs Calendar View)

```
public NoteAdapter(Activity activity, ArrayList<Note> notes) {
  this.activity = activity;
  this.allNotes = notes;
  inflater = (LayoutInflater) activity.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
}

@Override
public int getCount() {
  return this.allNotes.size();
}

@Override
public Object getItem(int position) {
  return this.allNotes.get(position);
}

@Override
public long getItemId(int position) {
  return this.allNotes.get(position).getId();
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
  View vi = convertView;
  vi = inflater.inflate(R.layout.note_item, null);

  Note currentNote = this.allNotes.get(position);
  TextView title = vi.findViewById(R.id.title_text_view);
  title.setText(currentNote.getTitle());

  TextView desc = vi.findViewById(R.id.desc_text_view);
  desc.setText(currentNote.getDescription());

  TextView created = vi.findViewById(R.id.date_created_text_view);
  created.setText(new SimpleDateFormat("dd MMM").format(currentNote.getDateCreated()));

  int backgroundColor = (position % 2 == 0) ?
        ContextCompat.getColor(activity.getBaseContext(), R.color.odd) :
        ContextCompat.getColor(activity.getBaseContext(), R.color.even);

  //vi.findViewById(R.id.rel_note).setBackgroundColor((backgroundColor));

  return vi;
}

public void clear() {
  this.allNotes.clear();
}

public void addAll(List<Note> allNotes) {
  this.allNotes.addAll(allNotes);
```

```
}
```

We implemented trigger from calendar to trigger adapter to show notes on specific date

```
private void updateUIWithDate(Date d) {
    ArrayList<Note> notes = db.getAllNotesByDate(d);
    DateFormat dateFormat = null;
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.N) {
        dateFormat = new SimpleDateFormat("MM DD YYYY");
    }
    String strDate = dateFormat.format(d);

    if (noteAdapter == null) {
        noteAdapter = new NoteAdapter(this, notes);
        //Toast.makeText(CalendarActivity.this, "No notes found on " +
makeDateString(d.getDay(),d.getMonth(),d.getYear()) , Toast.LENGTH_SHORT).show();
        notesListView.setAdapter(noteAdapter);
    } else {
        //Toast.makeText(CalendarActivity.this, "You have: " + notes.size() + " events." ,
Toast.LENGTH_SHORT).show();
        noteAdapter.clear();
        noteAdapter.addAll(notes);
        noteAdapter.notifyDataSetChanged();
    }

    TextView txt = findViewById(R.id.txt_noitems);
    if (notes.size() <= 0) {
        txt.setVisibility(View.VISIBLE);
    } else {
        txt.setVisibility(View.INVISIBLE);
    }

    notesListView.setVisibility(View.VISIBLE);
    CalendarView calendarView = findViewById(R.id.id_cal);
    calendarView.setVisibility(View.INVISIBLE);
}
```

3.3 Navigation

We used Bottom Bar Navigation and fragment to navigate between pages

```
private void setBottomNav() {
  BottomNavigationView bottomNav = findViewById(R.id.bottom_nav);
  bottomNav.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
      int id = item.getItemId();

      if (id == R.id.nav_add) {
        startActivity(new Intent(MainActivity.this, AddNoteActivity.class));
      }

      if (id == R.id.nav_cal) {
        startActivity(new Intent(MainActivity.this, CalendarActivity.class));
      }

      if (id == R.id.nav_search) {
        startActivity(new Intent(MainActivity.this, MainActivity.class));
      }

      return true;
    }
  });

}
```

3.3 Date Picker

We implement a widget for Date picker when adding notes

```
private void initDatePicker() {
   DatePickerDialog.OnDateSetListener dateSetListener = new DatePickerDialog.OnDateSetListener() {
      @Override
      public void onDateSet(DatePicker datePicker, int year, int month, int day) {
         dueDate = new Date(year, month, day);
         month = month + 1;
         String date = makeDateString(day, month, year);
         Toast.makeText(AddNoteActivity.this, "Date Selected:" + date, Toast.LENGTH_SHORT).show();
         dateButton.setText(date);
      }
   };

   Calendar cal = Calendar.getInstance();
   int year = cal.get(Calendar.YEAR);
   int month = cal.get(Calendar.MONTH);
   int day = cal.get(Calendar.DAY_OF_MONTH);


   int style = AlertDialog.THEME_HOLO_LIGHT;

   datePickerDialog = new DatePickerDialog(this, style, dateSetListener, year, month, day);
   //datePickerDialog.getDatePicker().setMaxDate(System.currentTimeMillis());

}

private String makeDateString(int day, int month, int year) {
   return getMonthFormat(month) + " " + day + " " + year;
}

private String getMonthFormat(int month) {
   if (month == 1)
      return "JAN";
   if (month == 2)
      return "FEB";
   if (month == 3)
      return "MAR";
   if (month == 4)
      return "APR";
   if (month == 5)
      return "MAY";
   if (month == 6)
      return "JUN";
   if (month == 7)
      return "JUL";
   if (month == 8)
      return "AUG";
   if (month == 9)
      return "SEP";
```

```
    if (month == 10)
        return "OCT";
    if (month == 11)
        return "NOV";
    if (month == 12)
        return "DEC";

    //default should never happen
    return "JAN";
}

public void openDatePicker(View view) {
    datePickerDialog.show();
}
```

3.4 Calendar

We implemented a calendar view for calendar select with adapter to trigger notes list and note view

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_calendar);
  notesListView = findViewById(R.id.cal_list_view);
  db = new NoteDatabaseHelper(this);
  //ArrayList<Note> notes = db.getAllNotes();
  //Toast.makeText(CalendarActivity.this, "Welcome, you have: " + notes.size() + " events in calendar." ,
Toast.LENGTH_SHORT).show();
  CalendarView calendarView = findViewById(R.id.id_cal);
  TextView txt = findViewById(R.id.txt_noitems);
  txt.setVisibility(View.INVISIBLE);
  ListView listView = findViewById(R.id.cal_list_view);
  listView.setVisibility(View.INVISIBLE);

  notesListView.setOnItemClickListener((parent, view, position, id) -> {
     Note note = (Note) parent.getItemAtPosition(position);
     Intent intent = new Intent(CalendarActivity.this, NoteActivity.class);
     intent.putExtra("NOTE_ID", note.getId());
     startActivity(intent);
  });

  calendarView.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {
     @Override
     public void onSelectedDayChange(@NonNull CalendarView view, int year, int month, int dayOfMonth) {
        Date date = new Date(year, month, dayOfMonth);
        updateUIWithDate(date);
     }
  });
```

```
private void updateUIWithDate(Date d) {
  ArrayList<Note> notes = db.getAllNotesByDate(d);
  DateFormat dateFormat = null;
  if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.N) {
     dateFormat = new SimpleDateFormat("MM DD YYYY");
  }
  String strDate = dateFormat.format(d);

  if (noteAdapter == null) {
     noteAdapter = new NoteAdapter(this, notes);
     //Toast.makeText(CalendarActivity.this, "No notes found on " +
makeDateString(d.getDay(),d.getMonth(),d.getYear()) , Toast.LENGTH_SHORT).show();
     notesListView.setAdapter(noteAdapter);
  } else {
     //Toast.makeText(CalendarActivity.this, "You have: " + notes.size() + " events." ,
Toast.LENGTH_SHORT).show();
     noteAdapter.clear();
     noteAdapter.addAll(notes);
     noteAdapter.notifyDataSetChanged();
  }
```

```
  TextView txt = findViewById(R.id.txt_noitems);
  if (notes.size() <= 0) {
     txt.setVisibility(View.VISIBLE);
  } else {
     txt.setVisibility(View.INVISIBLE);
  }

  notesListView.setVisibility(View.VISIBLE);
  CalendarView calendarView = findViewById(R.id.id_cal);
  calendarView.setVisibility(View.INVISIBLE);
}
```

3.5 Database Section

We implemented a DAO structure for SQLite Database

```
public class NoteDatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "NoteDatabase";
    private static final int DATABASE_VERSION = 1;
    private static final String TABLE_NAME = "Notes";
    private static final String COLUMN_ID = "id";
    private static final String COLUMN_TITLE = "title";
    private static final String COLUMN_DESCRIPTION = "description";
    private static final String COLUMN_DATE_CREATED = "dateCreated";

    public NoteDatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String createTable = "CREATE TABLE " + TABLE_NAME + " (" +
                COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                COLUMN_TITLE + " TEXT, " +
                COLUMN_DESCRIPTION + " TEXT, " +
                COLUMN_DATE_CREATED + " INTEGER)";
        db.execSQL(createTable);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }

    public void addNote(Note note) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_TITLE, note.getTitle());
        values.put(COLUMN_DESCRIPTION, note.getDescription());
        values.put(COLUMN_DATE_CREATED, note.getDateCreated().getTime());
        db.insert(TABLE_NAME, null, values);
        db.close();
    }

    public void deleteNote(int id) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_NAME, COLUMN_ID + " = ?", new String[]{String.valueOf(id)});
        db.close();
    }

    public Note getNote(int id) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.query(TABLE_NAME, new String[]{COLUMN_ID, COLUMN_TITLE,
COLUMN_DESCRIPTION, COLUMN_DATE_CREATED},
```

```
            COLUMN_ID + " = ?", new String[]{String.valueOf(id)}, null, null, null);
    if (cursor != null)
        cursor.moveToFirst();
    Note note = new Note(cursor.getInt(0), cursor.getString(1), cursor.getString(2));
    Date d = new Date(cursor.getLong(3));
    note.setDateCreated(d);
    cursor.close();
    return note;
  }

  public Note updateNote(Note note) {
    SQLiteDatabase db = this.getReadableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_TITLE, note.getTitle());
    values.put(COLUMN_DESCRIPTION, note.getDescription());
    values.put(COLUMN_DATE_CREATED, note.getDateCreated().getTime());
    int tru = db.update(TABLE_NAME, values, COLUMN_ID + " = ?", new
String[]{String.valueOf(note.getId())});

    return note;
  }
```

3.6 Notification Section (Disabled)

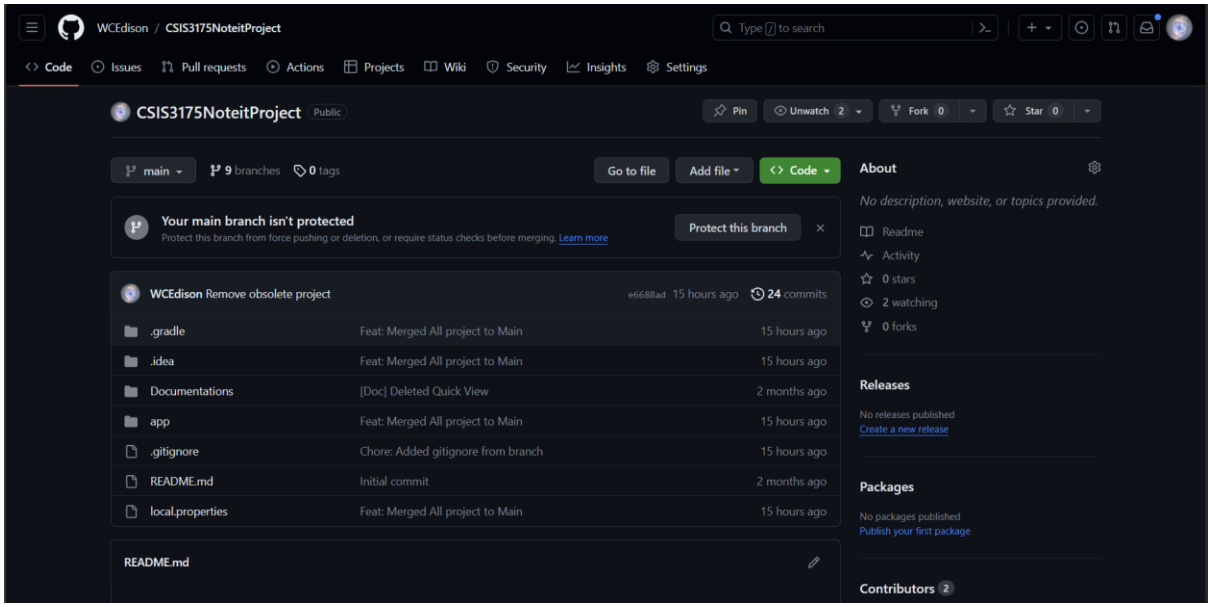
We implemented a notification include channel to display notification, but it is disabled as we have not implemented alarm system in the background thus the trigger will be incorrect

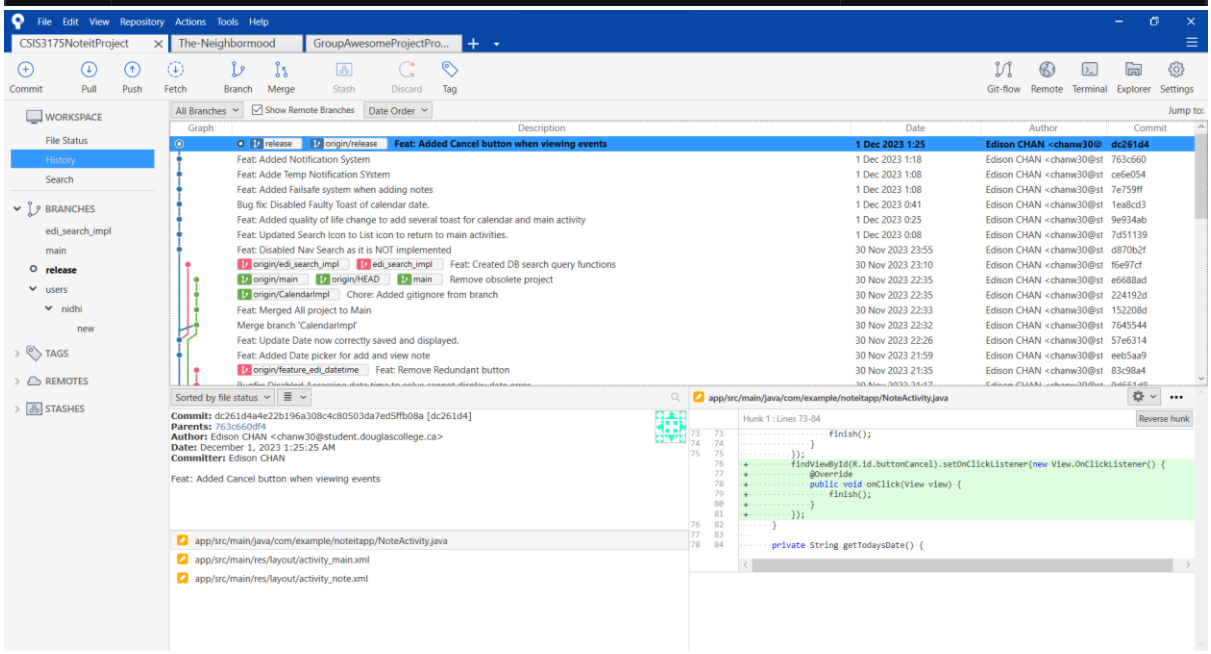
```
private void createNotificationChannel() {
    // Create the NotificationChannel, but only on API 26+ because
    // the NotificationChannel class is not in the Support Library.
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = getString(R.string.channel_name);
        String description = getString(R.string.channel_description);
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
        channel.setDescription(description);
        // Register the channel with the system; you can't change the importance
        // or other notification behaviors after this.
        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
}

void TriggerNotification(String title, String content) {
    // Create an explicit intent for an Activity in your app.
    Intent intent = new Intent(this, MainActivity.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG_IMMUTABLE);

    NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
            .setSmallIcon(R.drawable.notification_icon)
            .setContentTitle(title)
            .setContentText(content)
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
            // Set the intent that fires when the user taps the notification.
            .setContentIntent(pendingIntent)
            .setAutoCancel(true);

    NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);

    // notificationId is a unique int for each notification that you must define.
    if (ActivityCompat.checkSelfPermission(this, android.Manifest.permission.POST_NOTIFICATIONS) !=
PackageManager.PERMISSION_GRANTED) {
        // TODO: Consider calling
        //    ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        //   public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                          int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.
        return;
    }
    int id = (int) Calendar.getInstance().getTime().getTime();
    notificationManager.notify(id, builder.build());
```

3.7 Source Control

We used github for source control of the project
https://github.com/WCEdison/CSIS3175NoteitProject.git

The latest branch with release

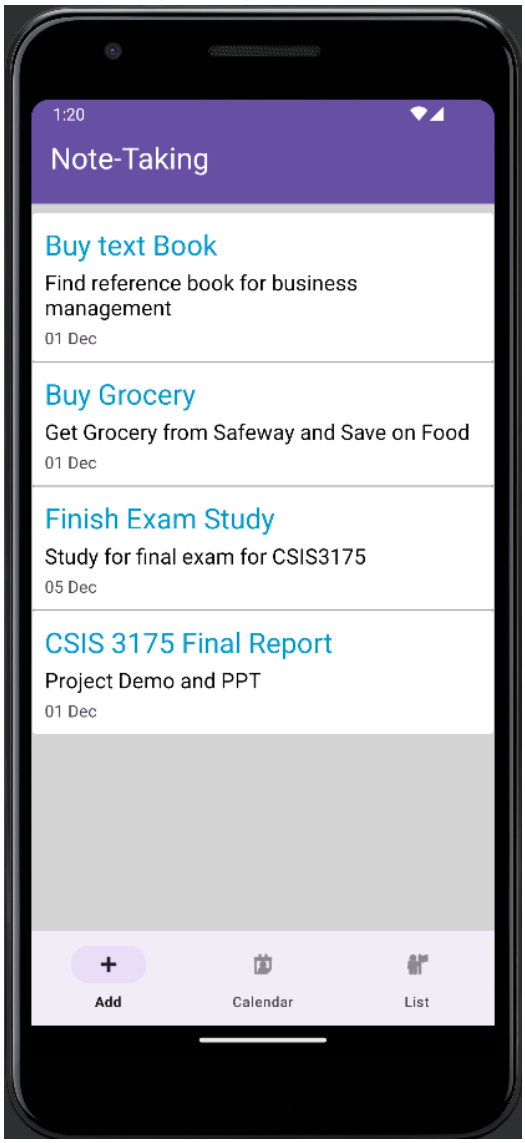




4. Testing chapter

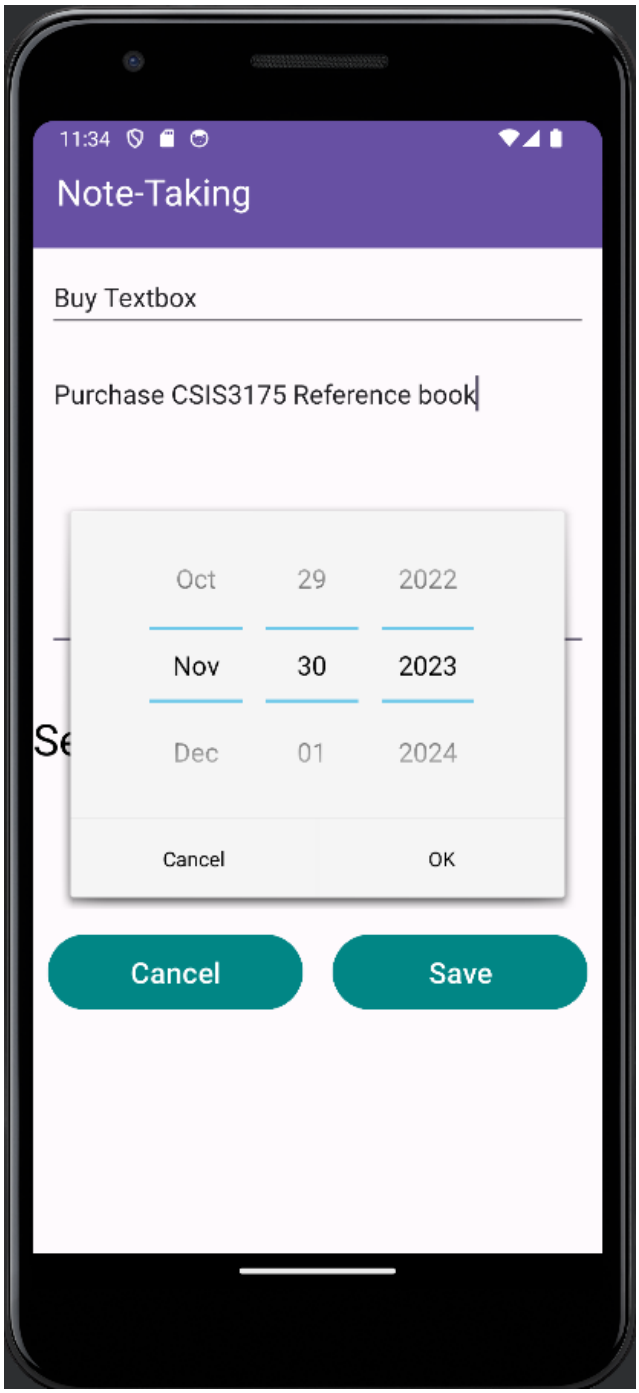
4.1 Main Page

We tested and we can view the list of items and also press add to create new items. Bottom bar can successfully navigate to other activities



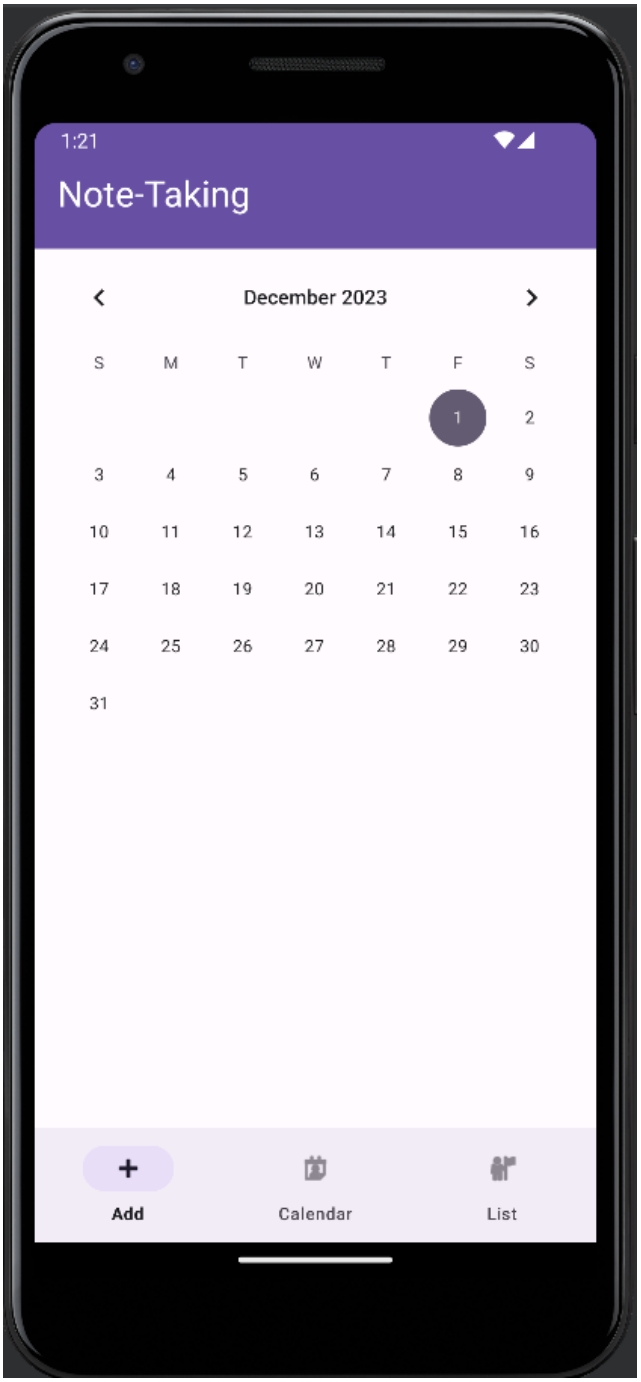
4.2 Note Add/Edit

We tested we can add persistent data, add text and edit date in Date Picker



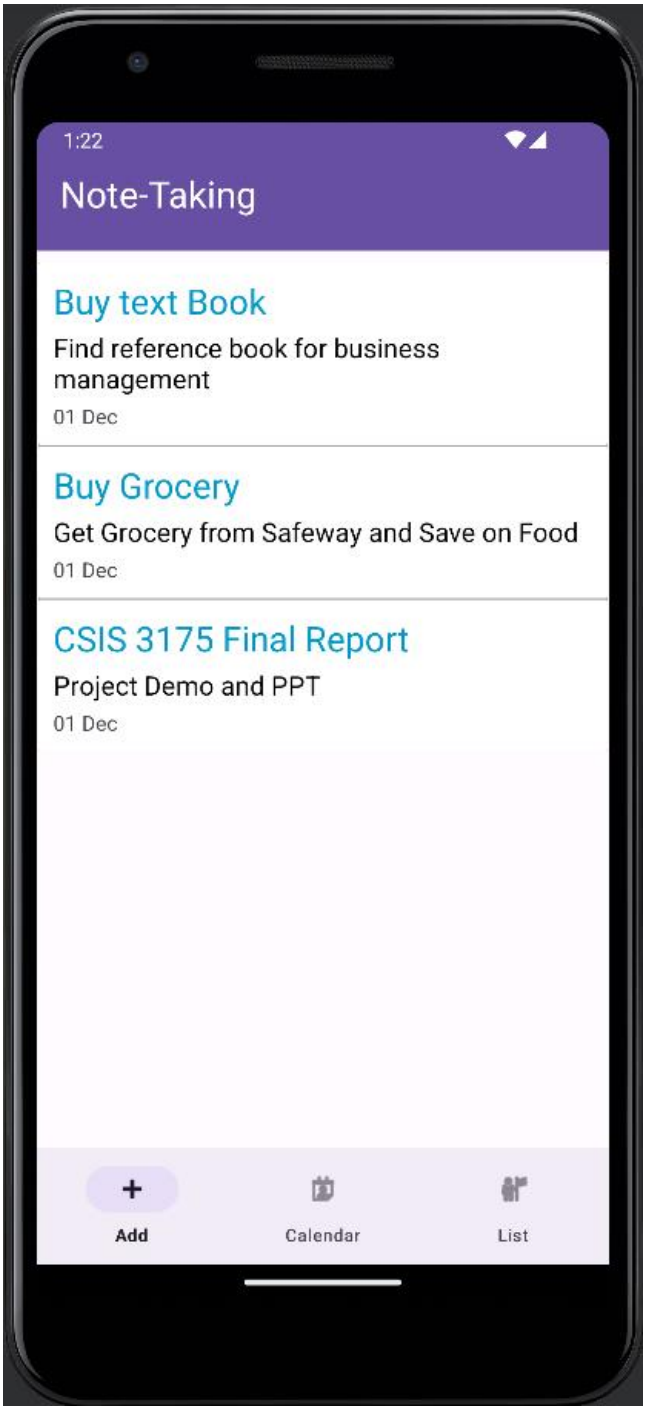
4.3 Calendar View

We tested that we can display a calendar, and trigger display list of task on specific date from Calendar



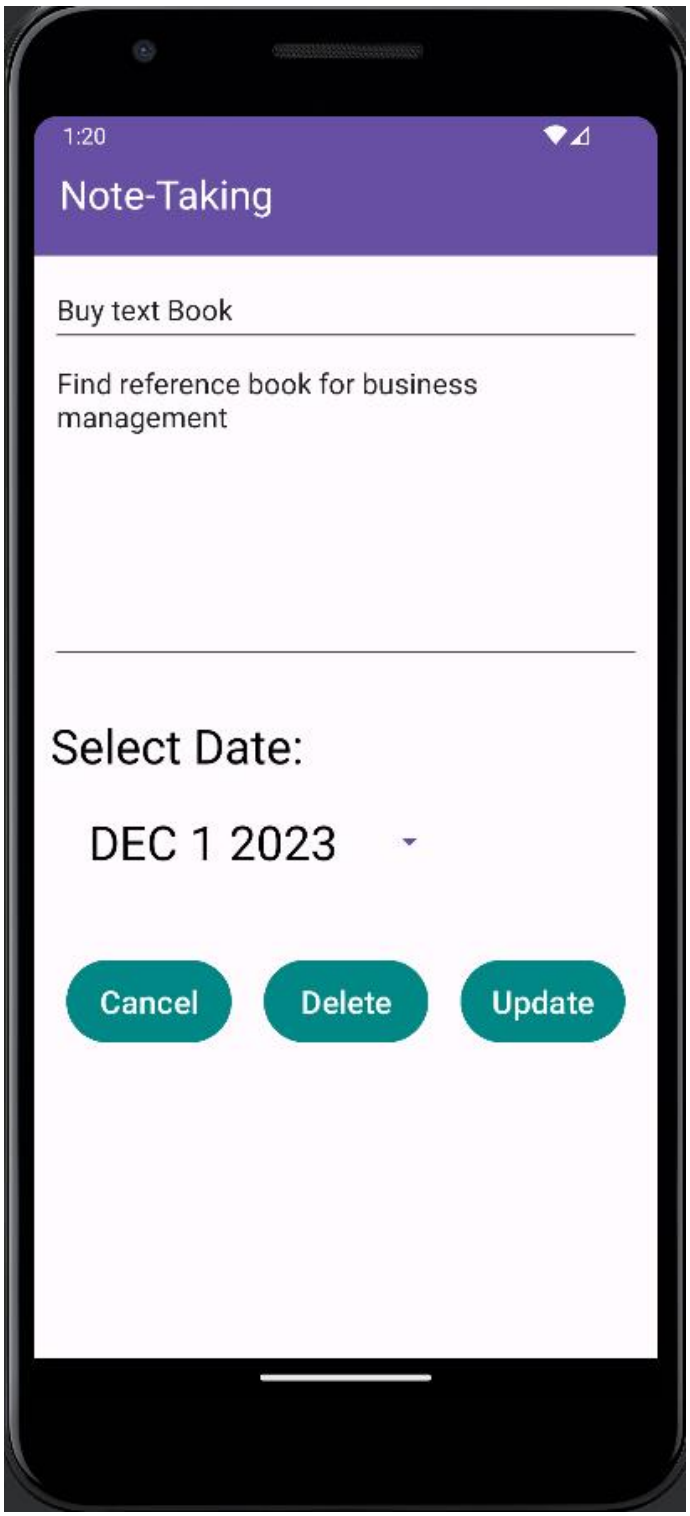
4.4 To do List by Date View

We tested that only specific task on the same due date is displayed, when redirecting when clicking on Dec 1



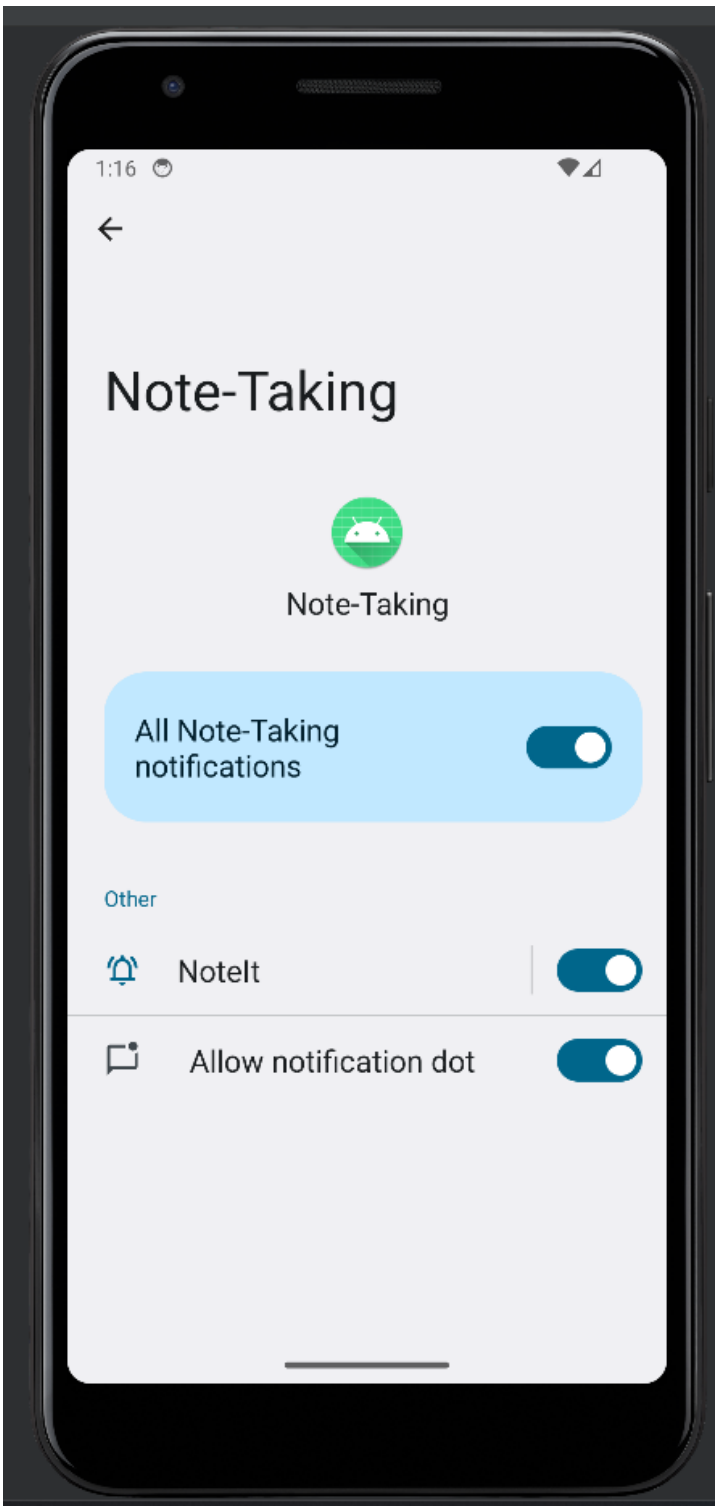
4.5 Update Notes, Note Adapter

We tested that after selecting the custom list, we can access the notes update screen, which is different from added new notes



4.6 Notification Triggers

We tested that our app has request notification channel and is ready to push notification when alarm is set up

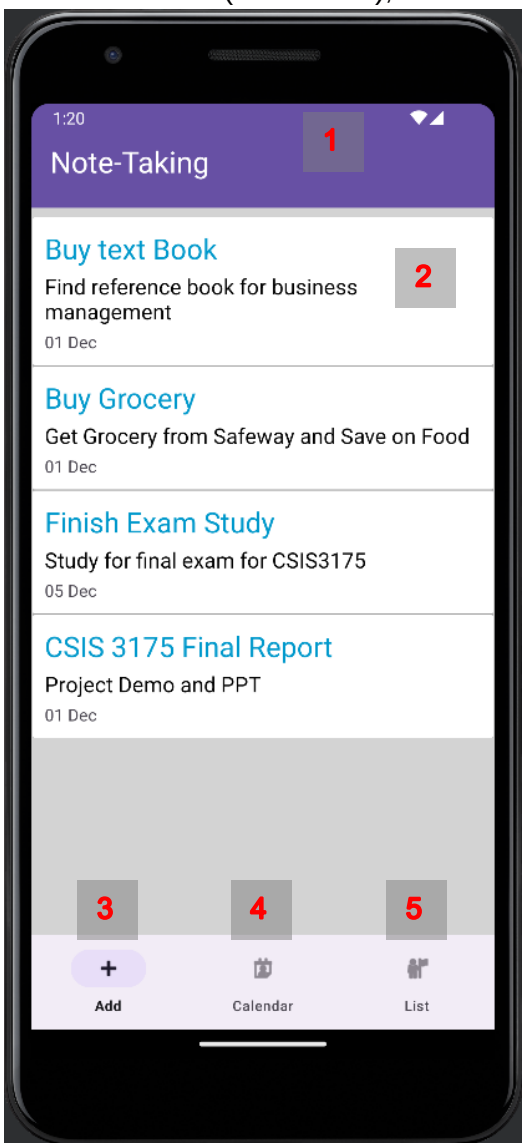


5. User guide chapter

5.1 Main Screen (to Do list)

Functions

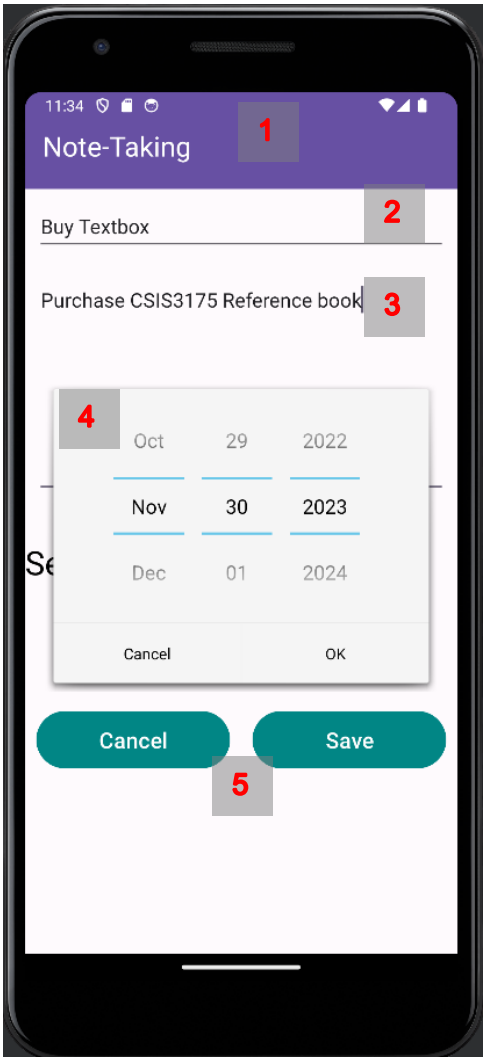
1. User can view the list of task on main screen
2. Each of the items can be pressed to enter update view
3. Pressing Add button at the bottom bar can create new notes
4. Pressing Calendar Bar at the bottom and go to calendar view
5. Pressing List will go to main screen (this screen), which is refreshing the screen



5.2 Add Notes

Functions

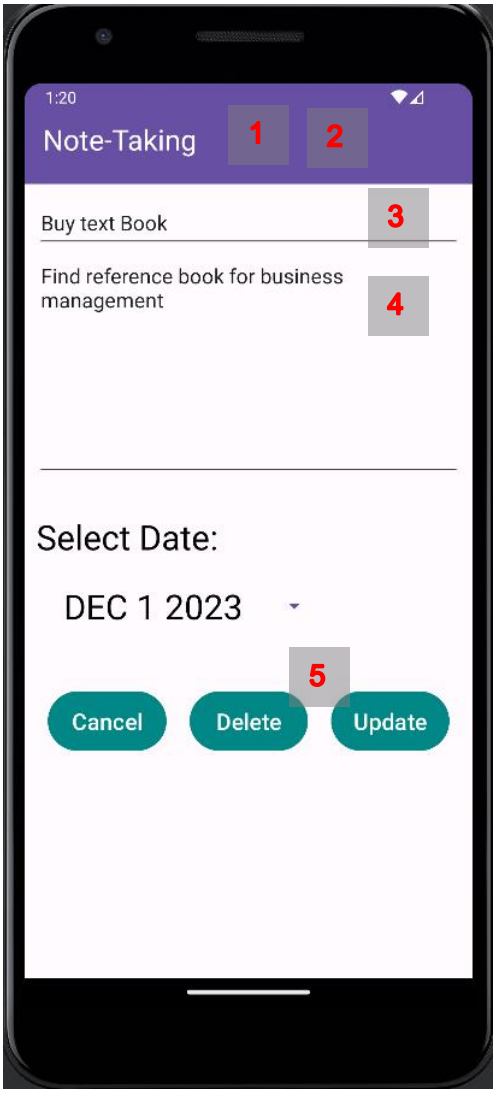
1. User can create note at this view
2. User can enter title at the header, title will always be display in to-do list
3. User can enter content at the middle bigger textbook, only the first few line will be displayed in to-do list for size limitation
4. Date picker widget will be used to select due date
5. User can press cancel to return to previous page, or save to save the new task
6. A hidden data, last edit will be stored for future use



5.3 Update Notes

Functions

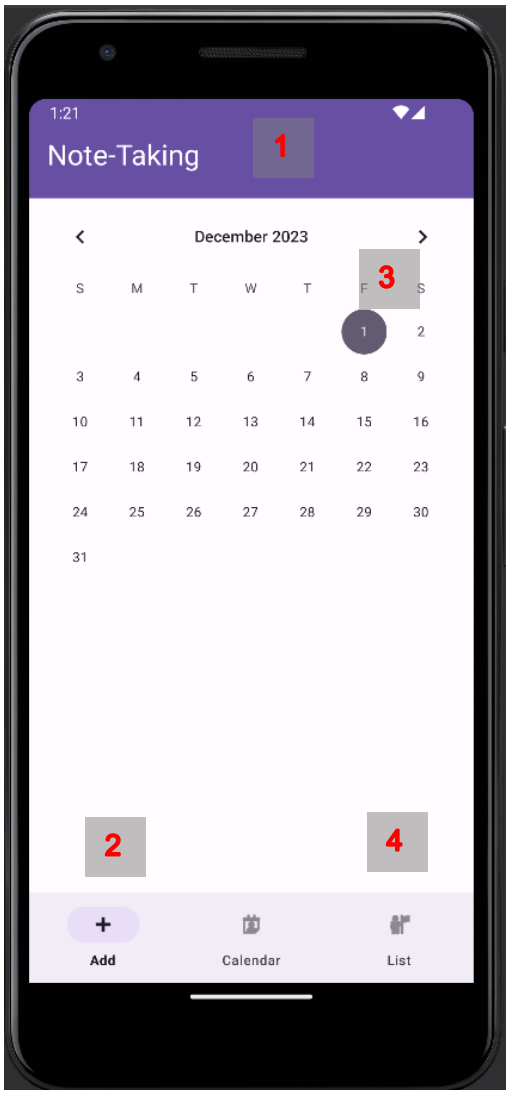
1. User can update note at this view
2. Unlike Add Notes, there are 2 version of this view, from Calendar via and from To do list, they have the SAME layout
3. User can update title at the header, title will always be display in to-do list
4. User can update content at the middle bigger textbook,
5. User can press Cancel to return to original page, delete to delete the task and update to save the changes



5.4 Calendar View

Functions

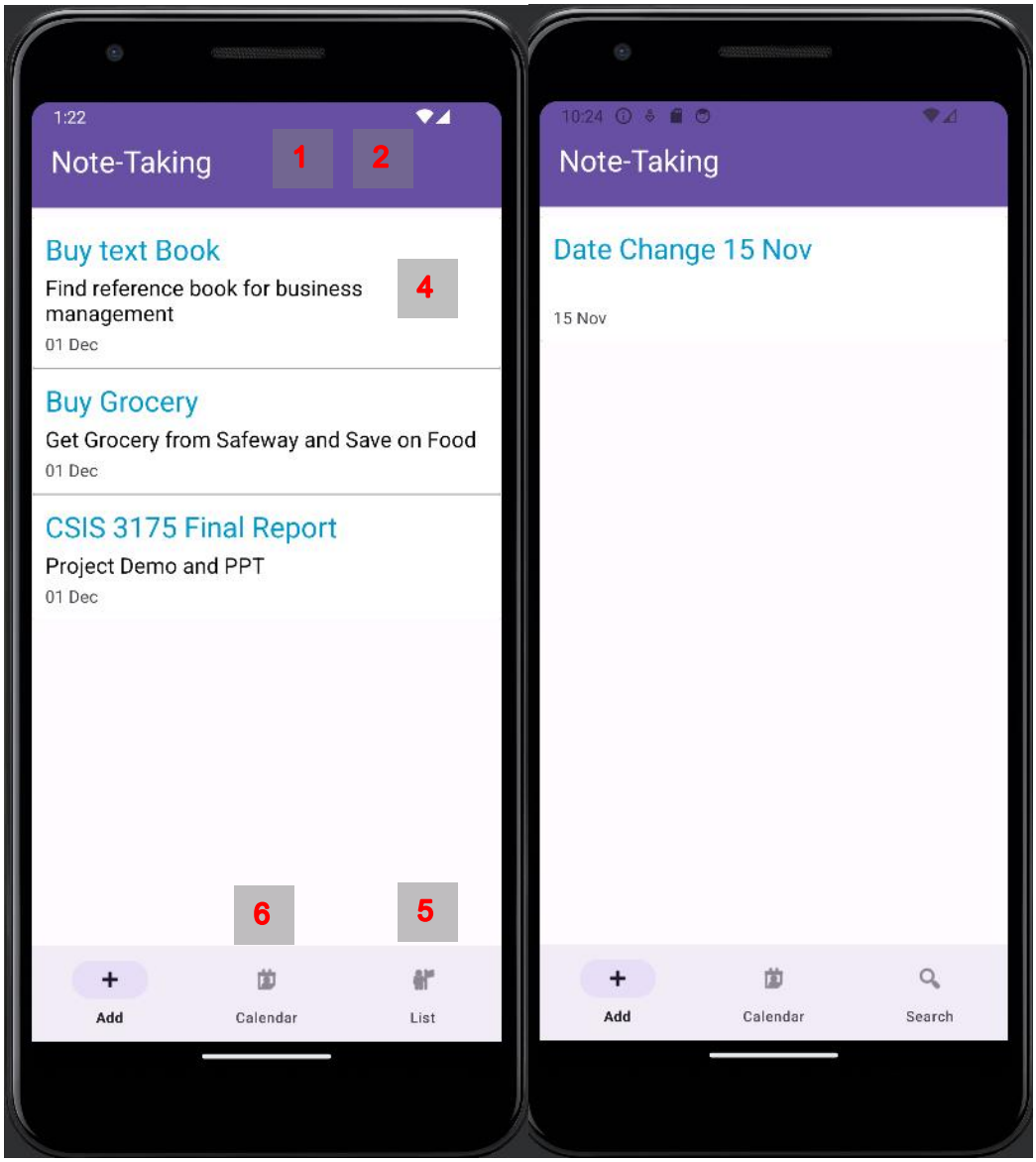
1. User can view calendar and select specific date for more detail
2. Pressing Add Notes will trigger the same UI for add notes
3. User can pick any date to view the custom task list
4. Pressing List will go to main screen



5.5 To do List by Date View

Functions

1. User can view calendar and select specific date for more detail
2. It has similar UI layout as the main screen, but is an entirely different activity
3. User can pick any date to view the custom task list
4. List will populate with tasks of that date, if nothing exists. Only the date will be shown
5. Pressing List will go to main screen
6. Pressing Calendar will go to calendar view



6. Conclusion section

The note-taking app is a useful tool for users who want to take notes on their Android devices. The app provides a variety of features that make it easy to create, edit, and organize notes. The app also provides users with the ability to search through their notes and customize the app's user interface. The calendar function is a useful addition that allows users to view notes due on a specific date. The app is built using Java and the Android Studio development environment and uses SQLite database to store notes.

In the future, we wish to be able to activate the push notification function and tie it back to our calendar view, thus we have an alarm notification when a task is due. We also hope to implement more data saved, like tag, image and text.

7. Reference

Add pickers to your app : android developers. Android Developers. (n.d.). https://developer.android.com/develop/ui/views/components/pickers

YouTube. (2021, December 19). *Make a notes app in Android Studio | room database | full tutorial*. YouTube. https://www.youtube.com/watch?v=Shh0N45S4hE

Calendar : android developers. Android Developers. (n.d.). https://developer.android.com/reference/android/icu/util/Calendar

Save data using sqlite : android developers. Android Developers. (n.d.). https://developer.android.com/training/data-storage/sqlite#java

Set up the app bar : android developers. Android Developers. (n.d.). https://developer.android.com/develop/ui/views/components/appbar/setting-up

Appcompatactivity : android developers. Android Developers. (n.d.-a). https://developer.android.com/reference/androidx/appcompat/app/AppCompatActivity

Working with the appbar : android developers. Android Developers. (n.d.). https://developer.android.com/guide/fragments/appbar#java

Working with the appbar : android developers. Android Developers. (n.d.-a). https://developer.android.com/guide/fragments/appbar