SMARTReminder

Project by Tudor-Neculai Bâlbă & Raul-Anton JAC

Group 1231EA

Content:

Introduction:

We made the SmartReminder app which is an android application used for reminders. You add a task, set a date & time for deadline and the app will send you a notification with 24h and 1h before the task deadline and the final notification when the event happens.

What we used?

We used the Android Studio IDE with Kotlin programming language and SQLite database.

File Structure:

- app
  - manifests
    - AndroidManifest.xml
  - kotlin+java
    - com.example.smartreminder
      - AboutActivity.kt
      - AddTask.kt
      - Database.kt
      - DeadlineNotifier.kt
      - DeadlineReceiver.kt
      - MainActivity.kt
      - TasksActivity.kt
      - SettingsActivity.kt
      - WeatherService.kt
  - res
    - drawable
      - ic_launcher_background.xml
      - ic_launcher_foreground.xml
      - ic_task_deadline.xml
      - ic_weather.xml
      - logo_ezgif_com_webp_to_jpg_converter.jpg
    - layout
      - activity_about.xml
      - activity_Add_task.xml
      - activity_main.xml
      - activity_settings.xml
      - activity_tasks.xml
    - mipmap
    - values
    - xml

Code explanations

1. Database
2. MainActivty
3. TasksActivity
4. AddTask
5. SettingsActivity
6. AboutActivity
7. WeatherService
8. DeadlineNotifier
9. DeadlineReceiver

---

1. **Database:**

We made our Database in SQLite (Database.kt).
To create the database we overrided the function onCreate with our database structure

ID	TASK	DEADLINE_DATE	DEADLINE_TIME

Functions:
- addTask
- deleteTask
- getAllTasks
- getTodayTasks(today)

2. **MainActivity**

In the MainActivity (Homepage) we have the following parts:
APIs:
- Quote API: Json API used with ZenQuotes https://zenquotes.io/api/random
  We 1st add the API url, then the HTTP request method (GET). then if we established the connection to the internet and with the server we continute to process the response. The next part of it it's basically what we find in the url and we saw a tutorial on YouTube for thsi.
- Weather API: Json API used with OpenWeather https://openweathermap.org/api
  Mostly the same as for the Quote API

For both api's we had to addthe INTERNET permission use in AndroidManifest.xml

Tasks for today – we display all tasks from today, if not, we display the message „No tasks today"

Notifications for tasks (24h before, 1h before and when the task is in progress – finished the deadline):
- 1st we get all the tasks from database
- then we call the DeadlineNotifier (more about it in the DeadlineNotifier explanations)

- in the for we retrve the taskDate and time STR comes from strings cause we tried with date and time format – a total fail – so we didn't changed the name of the variables back to taskDate and taskTime, we keep them as we used in testing mode
- sdf was used for formatting the date and time

  TRY:
  - we transform the taskDateTimeStr string into Date object, then we take the time from it to compute the time miliseconds to use it for comparisons
  - using the calendar we take the current date into nowCalendar and nowMillis for current time in miliseconds
  - then in the taskCalendar we put taskDateTime cause we will use taskDateTime later and we will modify it
  - we compute the time for notifications for each task for 24 hours in miliseconds and also 1h in miliseconds
  - and now we check if the they are for 24h or 1h notifications
    - for 24h we check the date also to be in this day (variabile isDayBefore)
    - for 1h we check the date to be today or one hour before if the day is also tommorrow
  - for notifications when deadline is done we simply check the time to be the same

Share button: you can share your daily program on any social app (like we did in the lab).

Bottom menu buttons – nothing special about them

function createNotificationChannel() is taken from notification module (Android Studio website)

3. <u>TasksActivity</u>

- See the tasks
- Delete tasks
- Add tasks

Code:
- 1st we connect the database
- then we implement the add task button
- and now we list the tasks
- in the if task!null we display the tasks and also we check if the user pressed long – this will popup the builder to display the alertdialog to check if the user is sure that want to delete the task

then we have the bottom fixed menu implementation like we had in mainactivity.kt

4. **AddTask**

- 1st we connect the database
- then we declare two variables which will store the selected date and time
- then we have the declaration of layout elements
- selectDateButton let the user choose the task date deadline and we verify the selected date:
    - if the month is smaller than 9 we add a 0 before. why? cause we will need when we display the tasks ordered by month; if we let it be 9, when we have 11, 11 will be displayed before 9 cause the date is a string
    - same for dayOfMonth like for month
- in the same way we did the selectTimeButton
- then after the addButton is clicked we process the adding into the database AFTER we check that all elements which will be inserted are NOT EMPTY.
- the backButton is declared as the buttons from the fixed bottom menu

then we have the bottom fixed menu implementation like we had in mainactivity.kt and all over the app

5. **SettingsActivity**

- we load the buttons for starting the foreground service (weather service)
- then we but the actions for each button with the corresponding functions from WeatherService.kt
- we use the themeSwitch button to switch the app mode (light or dark)
- we use the currentNightMode cause we had some problems with flickering and that's the only solution to prevent it by saving the last mode

then we have the bottom fixed menu implementation like we had in mainactivity.kt and all over the app

6. **AboutActivity**

In the about activity we have some details about the app, how to use it, all made in the activity_about.xml.

7. **WeatherService**

This part of our app will manage the weather forecast in the notification bar. How? Let's see:

- 1st we call the createNotificationChannel (taken from AndroidStudio website)
- we implement the foreground service (WeatherService) which is responsible for fetching and displaying the weather information. This service runs continuously in the background, and it's made a foreground service to keep it running reliably even when the app is in the background. A notification is shown to the user while the service is active

- we use a Timer object to set up a task that will execute every 60 seconds. The task calls the fetchWeather() method which is responsible for fetching the latest weather data
- In the fetchWeather() function, a coroutine (concept in Kotlin (and other programming languages) that allows you to write asynchronous, non-blocking code in a more readable and structured way) is launched on the IO dispatcher to perform the network request asynchronously. An HTTP GET request is made to the OpenWeatherMap API, fetching weather data for a specified city (Bucharest in this case). Upon a successful response, the data is parsed to retrieve the weather description and temperature, which are then used to update the lastWeatherUpdate string with the latest information
- we update the notification by calling updateNotification(lastWeatherUpdate). This updates the content of the ongoing notification with the most recent weather data, ensuring that the user is kept informed of the latest weather conditions. The notification is continuously updated every 60 seconds as long as the service is running, ensuring real-time data is displayed

8. **DeadlineNotifier**

The scheduleNotification function schedules an alarm to trigger a notification one minute before a task's deadline. It checks for permissions to schedule exact alarms on Android 12+ and requests them if not granted. The function calculates the time for the notification, creates an intent to trigger a DeadlineReceiver, and wraps it in a PendingIntent. It then uses AlarmManager to set the exact time for the notification. If successful, it logs the schedule; otherwise, it shows a Toast with an error message.

FUN FACT: We didn’t activated the notifications 1st cause we didn’t read the full page before do it and we did this part 3-4 times cause we thought that something in the code was wrong :)

9. **DeadlineReceiver**

We have a BroadcastReceiver that listens for alarms triggered by the DeadlineNotifier. When the alarm goes off, it receives the task name from the intent. It then creates a notification using NotificationCompat.Builder with a title indicating the task deadline and the task name. This notification is displayed with a high priority and a specified icon. The NotificationManager is used to issue the notification, and the task name's hash code is used to uniquely identify it.

What we used from the list of components?

1. Foreground Services: Weather notification in the notification bar
2. Background Services: all notifications that are preparing in the background, weather notification in the notification bar
3. Intents
4. Activities
5. Broadcast Receivers

6. Shared Preferences: light mode/dark mode
7. Content Providers: share your daily program
8. Database
9. Usage of external APIs: Weather and Quotes
10. Notifications – 24h, 1h, 0min:0sec notifications