Binaya Rimal

01/26/2025

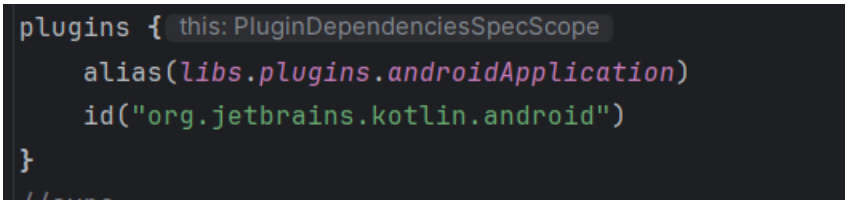
Software Design and Engineering

For the Software Design and Engineering portion of our project, I have been converting my code from Java to Kotlin. Once all the code is converted, we will begin addressing some of the issues encountered during the original development of the application, primarily the notification feature.

To facilitate the conversion, I first updated our Gradle configuration to include the Kotlin plugin, enabling our project to handle Kotlin files.

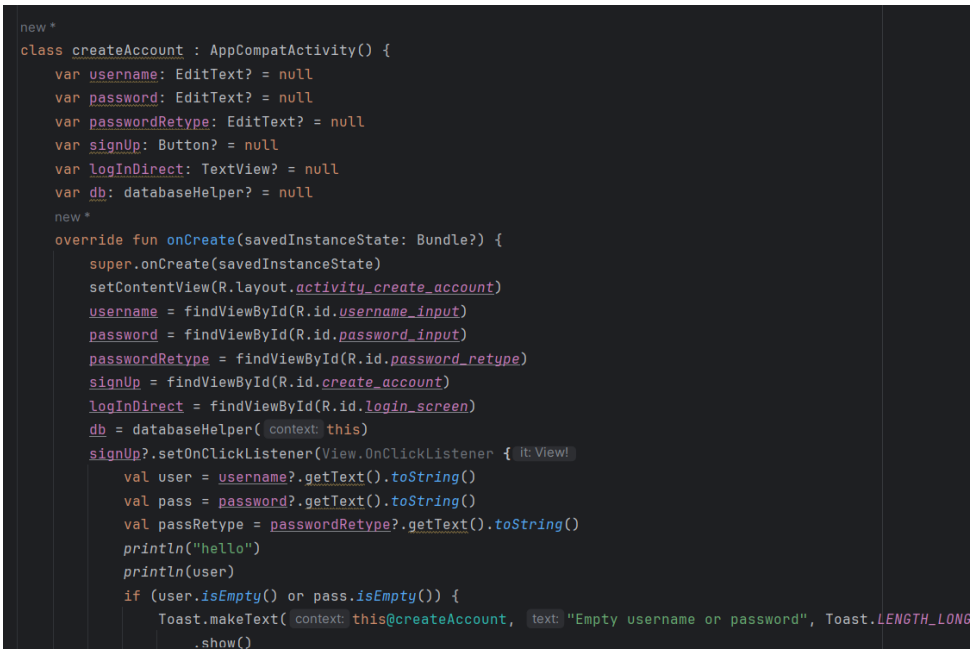
I then began the conversion process by researching Kotlin for Android development and systematically converting files one by one. One syntax difference that caused some challenges was Kotlin's use of nullable types, indicated by a question mark (?). Kotlin introduces several modern features, including null safety, extension functions, and a more concise syntax. Aside from these minor adjustments, the transition proceeded smoothly.

Here is an example of the converted code:



I prioritized converting the code before adding new features to ensure the existing functionality remains intact in the new language. This also guarantees compatibility with Kotlin-based libraries or frameworks that we may integrate in the future.

Below is an example of code converted from the Java to Kotlin:



```
new *
class createAccount : AppCompatActivity() {
    var username: EditText? = null
    var password: EditText? = null
    var passwordRetype: EditText? = null
    var signUp: Button? = null
    var logInDirect: TextView? = null
    var db: databaseHelper? = null
    new *
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_create_account)
        username = findViewById(R.id.username_input)
        password = findViewById(R.id.password_input)
        passwordRetype = findViewById(R.id.password_retype)
        signUp = findViewById(R.id.create_account)
        logInDirect = findViewById(R.id.login_screen)
        db = databaseHelper( context: this)
        signUp?.setOnClickListener(View.OnClickListener { it: View!
            val user = username?.getText().toString()
            val pass = password?.getText().toString()
            val passRetype = passwordRetype?.getText().toString()
            println("hello")
            println(user)
            if (user.isEmpty() or pass.isEmpty()) {
                Toast.makeText( context: this@createAccount,  text: "Empty username or password", Toast.LENGTH_LONG
                    .show()
```

Another major improvement in this iteration is the notification feature. The previous version of the app had issues with notifications, particularly the lack of real-time text notifications for same-day events. To address this, I implemented a feature that triggers real-time SMS notifications when an event occurs on the same day.

To achieve this, we now request SMS notification permissions upon login. This prompt appears only if the necessary permissions have not already been granted. The permission check and request are handled within the login file of our codebase.

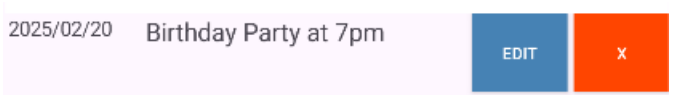
```
            // Check if SMS permission is granted
            if (ContextCompat.checkSelfPermission( context: this@Login, Manifest.permission.SEND_SMS) == PackageManager.PERMISSION_GRANTED) {
                startActivity(Intent( packageContext: this@Login, events::class.java))
                Toast.makeText( context: this@Login,  text: "Login Successful", Toast.LENGTH_LONG).show()
            } else {
                startActivity(Intent( packageContext: this@Login, allowNotification::class.java))
            }
        } else {
            Toast.makeText( context: this@Login,  text: "Account doesn't exist", Toast.LENGTH_LONG).show()
        }
    }
})
```

On the events page that lists all the events in the database, I check the date of each event and if the date matches with the current day(today) we send an SMS message reminding the user of the event. Below is the code send SMS function that sends the message.

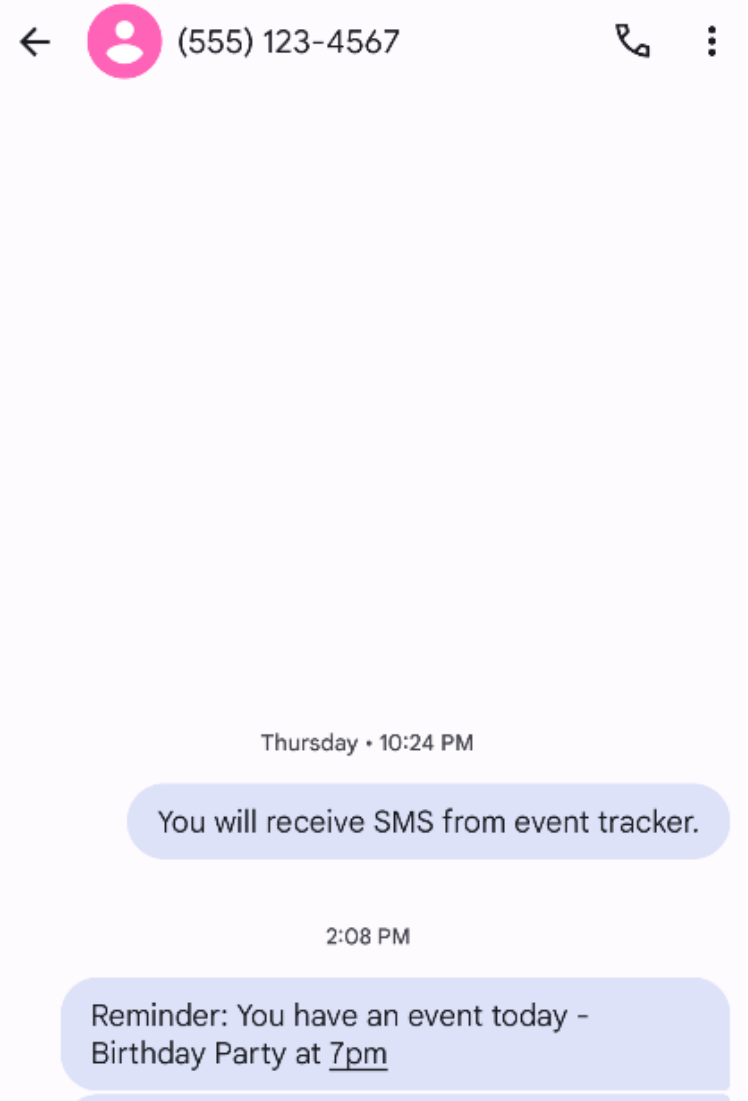
```
    Binaya Rimal
    private fun sendSms(phoneNumber: String, message: String) {
        if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.SEND_SMS) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions( activity: this, arrayOf(Manifest.permission.SEND_SMS),  requestCode: 1)
        } else {
            try {
                val smsManager = applicationContext.getSystemService(SmsManager::class.java)
                smsManager.sendTextMessage(phoneNumber,  scAddress: null, message,  sentIntent: null,  deliveryIntent: null)
                Toast.makeText( context: this,  text: "SMS Sent", Toast.LENGTH_LONG).show()
            } catch (e: Exception) {
                Toast.makeText( context: this,  text: "Failed to send SMS", Toast.LENGTH_LONG).show()
            }
        }
    }

    Binaya Rimal
```

One of the events that match the current date:



Message sent to the user:



From these enhancements, I learned software engineering and design principles, including code maintainability and gradual migration between technologies by systematically converting Java code to Kotlin while making sure functionality remained intact. I gained insight into best practices for software development, such as prioritizing stability before introducing new features, which is crucial for scalable and maintainable systems.

Additionally, I improved my understanding of feature enhancement and debugging, particularly in resolving issues with real-time notifications. I improved the User Experiences

by implementing the SMS notification system allowing users to get real time message notifications.

Below is the video demo that showcases what was done during this enhancement:

https://www.youtube.com/watch?v=Rv657tdYEnA

Instructions to download the app:

Pre work:

Download Android Studio

Clone the Repository

Open the Project in Android Studio

1. Open **Android Studio** on your laptop.
2. Click **"Open"** or **"Open an Existing Project"**.
3. Navigate to the project folder you extracted/downloaded.
4. Click **OK** and wait for Android Studio to load the project.
5. **Wait for Gradle to sync** – If prompted, click **"Sync Now"**.
6. If you see a **missing SDK error**, go to **File > Project Structure > SDK Location** and set the correct Android SDK path.
7. If necessary, update dependencies in build.gradle and click **"Sync Now"** again.
8. Connect a **physical Android device** via USB **or** use an **Android Emulator** (AVD).

9. Click **Run**

10. Select the target device/emulator and wait for the app to launch.