CSE225 ETP REPORT

on the topic

Stock Market Watchlist App

Abhishek Haridasan

12012647

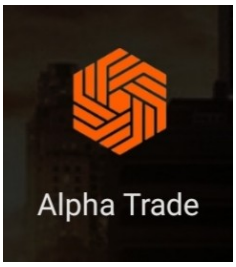
INDEX

INTRODUCTION

ALPHA  TRADE

It is a stock watchlist application. There is a fixed amount of pre-determined stocks that the app supports, where most of them are stocks in NASDAQ and NYSE. The user can add or remove stocks from their watchlist, see the description of a particular stock, see the graph of daily average price of a stock for the past 50 days, set a reminder to check on a particular stock.

Splash Screen



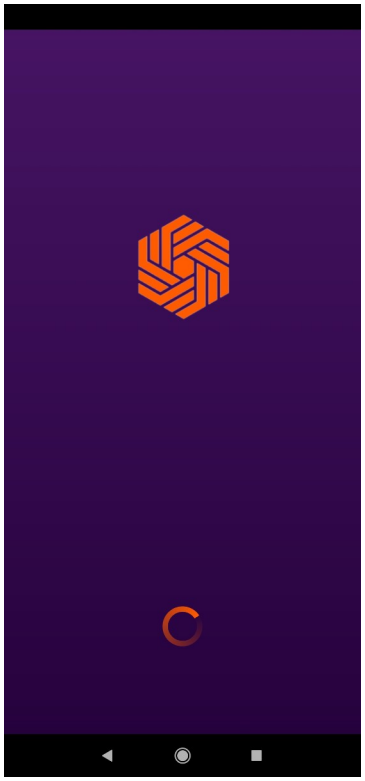
App logo and name

The app opens with the splash screen, which displays the logo and a indeterminate progressbar. The splash screen will run for 5 seconds. It also checks whether any user is already logged in or not using data from shared preferences. If someone is already logged in then intent is made directly for the HomePage else the user is sent to the LoginPage.

```
if (!loggedInSharedPreferences.contains("login")) {

    loggedInSharedPreferences.edit().putString("login", loginDefault["login"]).apply()
}

val loggedInData = loggedInSharedPreferences.getString( key: "login",  defValue: "")!!.split( ...delimiters: ", ").toMutableList()
```

```
if (loggedInData[0].toBoolean()) {
    startAppIntent = Intent( packageContext: this, HomePage::class.java)
    startAppIntent.putExtra( name: "UserID", loggedInData[1])
}
else {
    startAppIntent = Intent( packageContext: this, LoginPage::class.java)
}
```

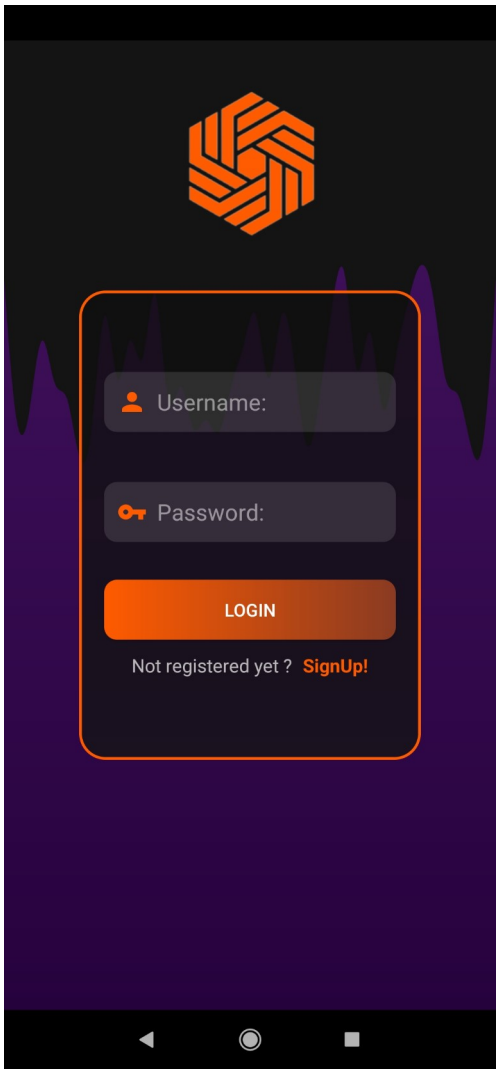
Code snippet for checking whether someone is currently logged in



Splash Screen

Login Page

The login page has a constraint layout as the root layout. The login credentials are verified by comparing
it with data from shared preferences. If they matches an intent for the HomePage is created and username is passed as a string extra. The shared preferences is also edited to show that the user is currently logged in, so that they don't have to log in again next time unless they sign out. On clicking the "SignUp!" text view, user is taken to the SignUpPage.



Login Page

Sign Up Page

Here apart from the basic details, the user is allowed to upload a profile picture. On enabling the switch an intent for camera is created and the user can capture their profile picture. The captured image is made rounded using a ShapableImageView.

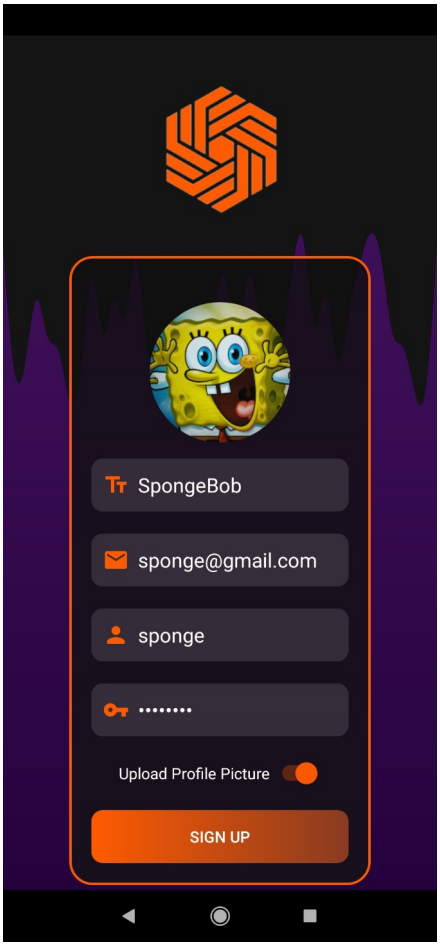
```
fun saveImage(imageView: ImageView , context: Context){

    val sharedPref = context.getSharedPreferences( name: "ImageData", Context.MODE_PRIVATE)
    val baos = ByteArrayOutputStream()
    val bitmap = imageView.drawable.toBitmap()
    val key = user_id.text.toString()
    bitmap.compress(Bitmap.CompressFormat.PNG , quality: 100,baos)
    val encodeImage= Base64.encodeToString(baos.toByteArray() , Base64.DEFAULT)
    with(sharedPref.edit()){ this: SharedPreferences.Editor!
        putString(key, encodeImage)
        apply()
    }
}
```

Function to convert the image into string and save it into shared preferences

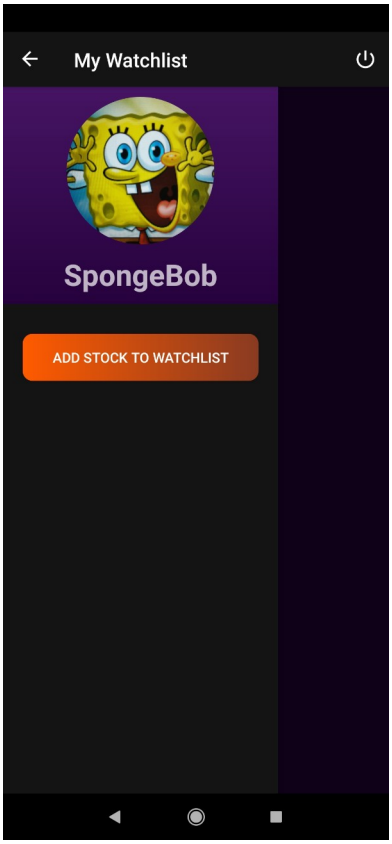
On clicking the signup button a new user with all the given details is created in the shared preferences. Then the user is taken back to the login page.

HomePage

The root layout here is a DrawerLayout as I had to create a navigation drawer. The username is retreived from the intent and the name and profile picture of the user is displayed in the navigation bar. A frame layout is also added to easily accommodate fragments.



SignUp Page

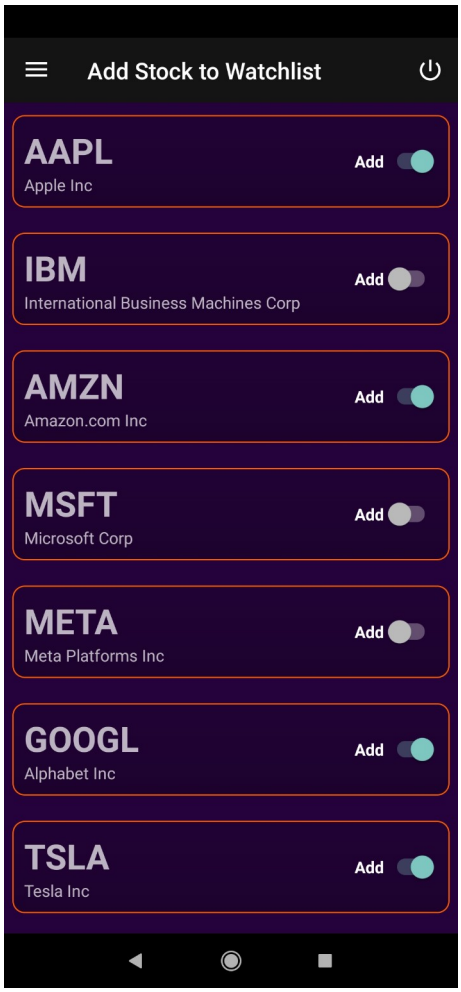


User details displayed in home page's navigation drawer

In the onCreate function of HomePage activity, the WatchListPage fragment is added to the frame layout.
A sign out button is also added in the action bar. A button to allow the user to edit their watchlist is added in the navigation drawer.

Add stock to watchlist Page

This page is a fragment, it dynamically adds custom views into a scroll view. Each custom view represents a stock in the list of about 15 stocks that the app supports. The custom view displays the stock symbol, the name of the company and a switch to add or removve the stock from the watchlist. Given the stock symbol all details about the stock are retreived using Alpha Vantage and Finnub API.



Add stock to watchlist page

```
fun getNameAPI(stockSymbol: String, apiKey: String, callback: (String) -> Unit) {

    val client = OkHttpClient()

    val url = "https://finnhub.io/api/v1/stock/profile2?symbol=$stockSymbol&token=$apiKey"

    val request = Request.Builder()
        .url(url)
        .build()

    client.newCall(request).enqueue(object : Callback {
        override fun onResponse(call: Call, response: Response) {
            val json = response.body()?.string()
            val data = JSONObject(json)
            //Log.d("TAG", data.toString())
            val companyName = data.optString( name: "name").trim()

            Handler(Looper.getMainLooper()).post {
                callback(companyName)
            }
        }

        override fun onFailure(call: Call, e: IOException) {
            e.printStackTrace()
        }
    })
}
```

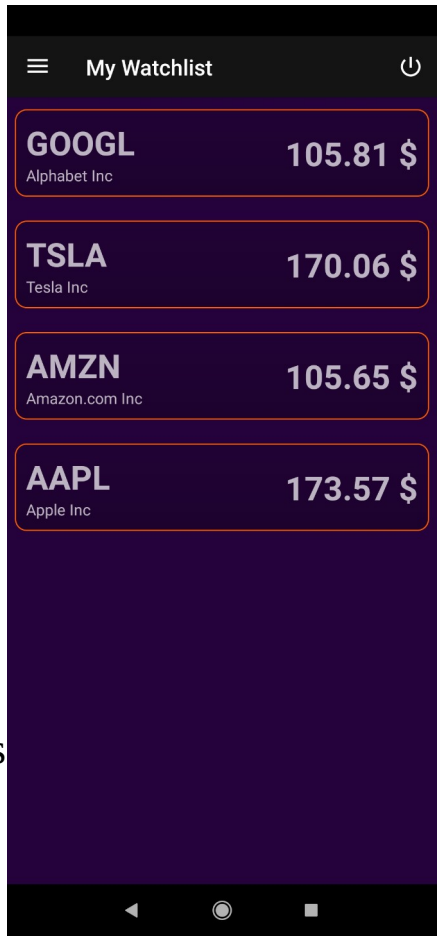
Function to retreive name of the company using finnub API

```
for (stock in defaultStocks) {

    val customView = CustomViewAddStock(requireContext(), userID, stock, apiKey)
    scrollView.addView(customView, layoutParams)
}
```

Code snippet of adding custom views dynamically

Watchlist Page

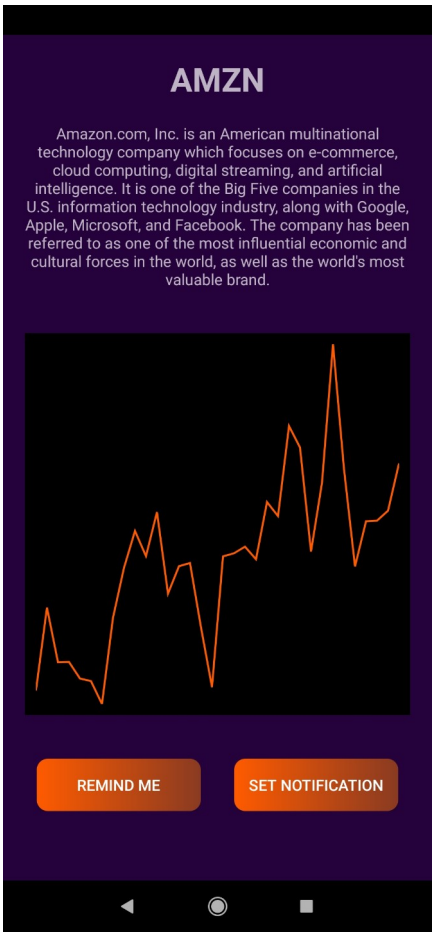
This page is also a fragment. It dynamically adds custom views into a scroll view for every stock in the user's watchlist. The custom view in this page displays the current price of the stock instead of a switch. All the custom views are assigned an onClickListener to redirect the user to the StockDisplayPage which shows more details about that particular stock.



Watchlist Page

Stock Display Page

On clicking a custom view in watchlist page, the user is redirected to this page which shows more details about the stock. It shows a description of the company and a line graph showing the daily price of the stock for past 50 days.



Stock Display Page

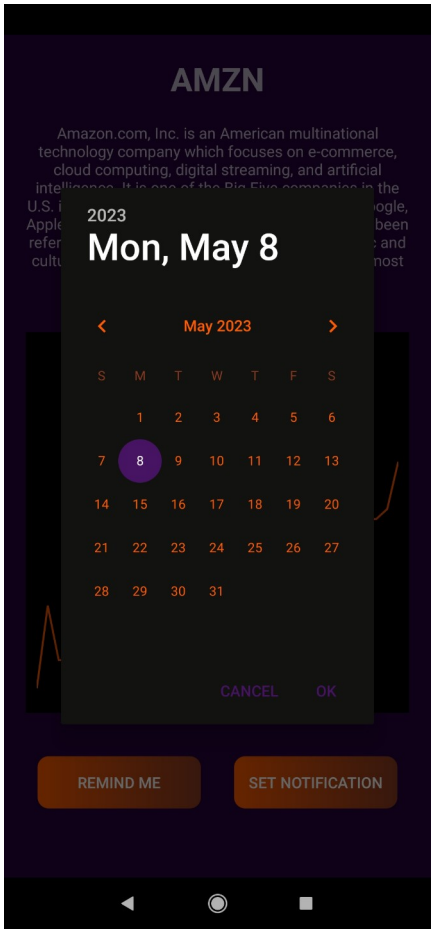
The graph is plotted by creating a custom view and using it's onDraw method.

A button to remind the user to check back on the stock is provided on this page. On clicking it, it opens a DatePicker and TimePicker dialog box and sets a

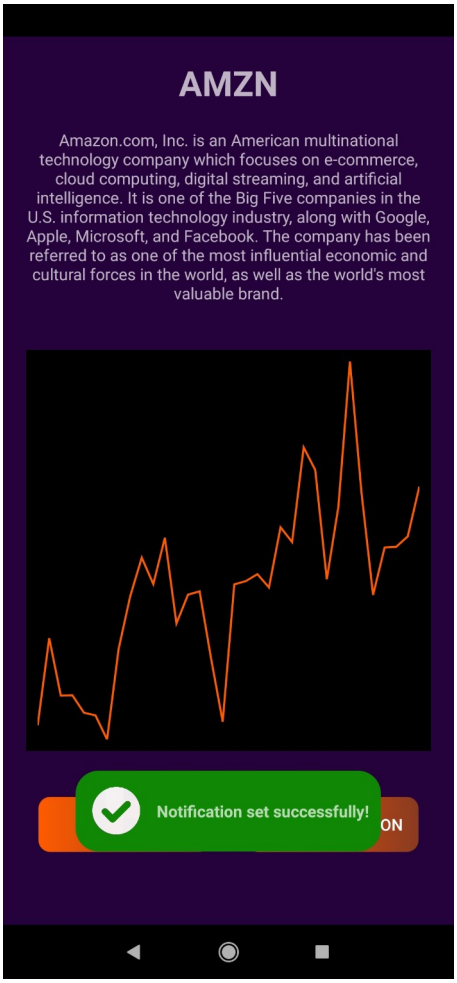
```
for ((i, value) in priceList.withIndex()) {

    val x = i * width.toFloat() / (priceList.size - 1)
    val y = (((value-minValue) / valueRange) * height)
    Log.d( tag: "Graph",  msg: "$i  $y $value")
    if (i == 0) {
        path.moveTo(x,  y: height-y)
    } else {
        path.lineTo(x,  y: height-y)
    }
}

canvas?.drawPath(path, paint)
```

Code snippet of plotting the graph

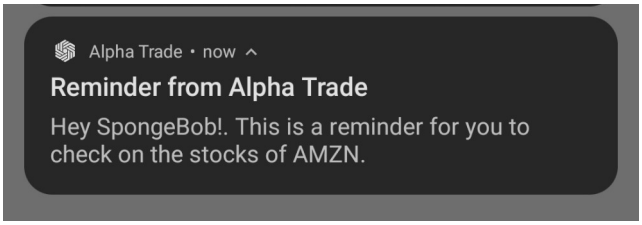
notification during the selected date and time using an AlarmManager. On successful completion a custom toast message is displayed.



DatePicker dialog box



Custom toast message



Reminder Notification

```
fun notify(view: View) {

    val vg:ViewGroup? = findViewById(R.id.custom_toast)
    val inflater = layoutInflater

    val layout: View = inflater.inflate(R.layout.custom_toast_layout,vg)
    layout.findViewById<TextView>(R.id.txt_toast).text = "Notification set successfully!"

    val toast = Toast(applicationContext)

    toast.duration = Toast.LENGTH_LONG
    toast.setView(layout)
    toast.show()
}
```

Function creating a custom toast

```
val builder = NotificationCompat.Builder(context, channelId: "my_channel_id")
    .setContentTitle("Reminder from Alpha Trade")
    .setContentText("Hey $name!. This is a reminder for you to check on the stocks of $stockSymbol.")
    .setSmallIcon(R.drawable.logo)
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)
    .setStyle(NotificationCompat.BigTextStyle().bigText( cs: "Hey $name!. This is a reminder for you to chec

notificationManager.notify( id: 1, builder.build())
```

A snippet of code of creating a notification in the BroadcastReceiver classs that handle the alarm

UNIT WISE TOPICS COVERED

- **Unit 1 –** Scroll View, Splash Screen, Progress Bar, Custom Toast
- **Unit 2 –** Intent, Pending Intents, Alarm Manager, Notification
- **Unit 3 –** Fragments, Dynamic addition, DatePicker, TimePicker
- **Unit 4 –** Custom Views, drawing on canvas
- **Unit 5 –** Shared Preferences
- **Unit 6 –** Navigation Drawer, DrawerLayout