PennyAndroid/activity_main.xml (Page 1 of 1)

```
<?xml version="1.0" encoding="utf-8"?>
<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="wrap_content"
   tools:context=".MainActivity">

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical"
      tools:layout_editor_absoluteX="16dp"
      tools:layout_editor_absoluteY="2dp">

      <EditText
         android:id="@+id/editText"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:hint="Enter an author"
         android:inputType="text" />

      <TextView
         android:id="@+id/textView"
         android:layout_width="match_parent"
         android:layout_height="wrap_content" />
   </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```


PennyAndroid/MainActivity.java (Page 1 of 3)

```
//---------------------------------------------------------------------------
// MainActivity.java
// Author: Bob Dondero
//---------------------------------------------------------------------------

package edu.Princeton.penny;

import android.widget.EditText;
import android.app.Activity;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.widget.TextView;
import android.text.TextWatcher;
import android.text.Editable;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import javax.net.ssl.HttpsURLConnection;
import java.util.Timer;
import java.util.TimerTask;

import org.json.JSONObject;
import org.json.JSONArray;

public class MainActivity extends Activity
{
   private EditText editText;
   private TextView textView;
   private AuthorSearchRunnable authorSearchRunnable = null;
   private Timer timer = null;

   //---------------------------------------------------------------------------

   private class AuthorSearchRunnable implements Runnable
   {
      private String author;
      private boolean shouldStop = false;

      AuthorSearchRunnable(String author)
      {
         this.author = author;
      }

      public void setShouldStop()
      {
         shouldStop = true;
      }

      public void run()
      {
         String books;

         try
         {
            URL url = new URL("https://pennyjson.onrender.com" +
                              "/searchresults?author=" + author);
            HttpsURLConnection con =
               (HttpsURLConnection) url.openConnection();

            con.setRequestMethod("GET");
            con.setDoInput(true);
            con.connect();
            int responseCode = con.getResponseCode();
```

PennyAndroid/MainActivity.java (Page 2 of 3)

```
66:             if (responseCode != HttpsURLConnection.HTTP_OK)
67:                throw new IOException(
68:                   "HTTP error code: " + responseCode);
69:
70:             // Read the response, a JSON document containing books,
71:             // from the server.
72:             BufferedReader in =
73:                new BufferedReader(
74:                   new InputStreamReader(con.getInputStream()));
75:             String jsonDoc = in.readLine();
76:             in.close();
77:
78:             // Use the JSON response to build a String, and assign
79:             // it to books.
80:             JSONArray jsonArray = new JSONArray(jsonDoc);
81:             StringBuilder booksBuilder = new StringBuilder();
82:             for (int i = 0; i < jsonArray.length(); i++)
83:             {
84:                JSONObject jsonObject = jsonArray.getJSONObject(i);
85:                booksBuilder.append(jsonObject.getString("isbn"));
86:                booksBuilder.append(": ");
87:                booksBuilder.append(jsonObject.getString("author"));
88:                booksBuilder.append(": ");
89:                booksBuilder.append(jsonObject.getString("title"));
90:                booksBuilder.append("\n");
91:             }
92:             books = booksBuilder.toString();
93:
94:             if (! shouldStop)
95:                runOnUiThread(() -> { textView.setText(books); });
96:          }
97:
98:          catch (Exception e)
99:          {
100:            if (! shouldStop)
101:               runOnUiThread(() -> { textView.setText(e.toString()); });
102:         }
103:      }
104:   }
105:
106:   //-----------------------------------------------------------------
107:
108:   private class MyTimerTask extends TimerTask
109:   {
110:      public void run()
111:      {
112:         String author = editText.getText().toString();
113:
114:         if (authorSearchRunnable != null)
115:            authorSearchRunnable.setShouldStop();
116:
117:         authorSearchRunnable = new AuthorSearchRunnable(author);
118:         Thread authorSearchThread = new Thread(authorSearchRunnable);
119:         authorSearchThread.start();
120:      }
121:   }
122:
123:   //-----------------------------------------------------------------
124:
125:   private class MyTextWatcher implements TextWatcher
126:   {
127:      public void onTextChanged(CharSequence s, int start,
128:                                int before, int count)
129:      {
130:      }
```

PennyAndroid/MainActivity.java (Page 3 of 3)

```
131:
132:      public void beforeTextChanged(CharSequence s, int start,
133:                                    int count, int after)
134:      {
135:      }
136:
137:      public void afterTextChanged(Editable s)
138:      {
139:         if (timer != null)
140:            timer.cancel();
141:         timer = new Timer();
142:         timer.schedule(new MyTimerTask(), 500);
143:      }
144:   }
145:
146:   //-----------------------------------------------------------------
147:
148:   protected void onCreate(Bundle activityState)
149:   {
150:      super.onCreate(activityState);
151:
152:      // Set the content view of the window to activity_main,
153:      // as defined in activity_main.xml.
154:      setContentView(R.layout.activity_main);
155:
156:      // Find the EditText object.
157:      editText = findViewById(R.id.editText);
158:
159:      // Instantiate a TextWatcher object, and install it as a
160:      // "text changed" listener on the EditText object. Subsequently,
161:      // immediately after the user changes the text within the
162:      // EditText object, the afterTextChanged() method will be called.
163:      editText.addTextChangedListener(new MyTextWatcher());
164:
165:      // Find the TextView object.
166:      textView = findViewById(R.id.textView);
167:
168:      // Allow the TextView object to scroll, just in case there are
169:      // many books.
170:      textView.setMovementMethod(new ScrollingMovementMethod());
171:   }
172: }
```

PennyAndroid/AndroidManifest.xml (Page 1 of 1)

```
 1: <?xml version="1.0" encoding="utf-8"?>
 2: <manifest xmlns:android="http://schemas.android.com/apk/res/android"
 3:    package="edu.Princeton.penny">
 4:
 5:    <uses-permission android:name="android.permission.INTERNET"/>
 6:    <uses-permission
 7:       android:name="android.permission.ACCESS_NETWORK_STATE"/>
 8:
 9:    <application
10:       android:allowBackup="true"
11:       android:icon="@mipmap/ic_launcher"
12:       android:label="@string/app_name"
13:       android:roundIcon="@mipmap/ic_launcher_round"
14:       android:supportsRtl="true"
15:       android:theme="@style/Theme.PennyAndroid">
16:       <activity android:name=".MainActivity" android:exported="true">
17:          <intent-filter>
18:             <action android:name="android.intent.action.MAIN" />
19:             <category android:name="android.intent.category.LAUNCHER" />
20:          </intent-filter>
21:       </activity>
22:    </application>
23: </manifest>
```

blank (Page 1 of 1)

```
1: This page is intentionally blank.
```

PennyIos/PennyIosApp.swift (Page 1 of 1)

```
 1: //
 2: //  PennyIosApp.swift
 3: //  PennyIos
 4: //
 5: //  Created by rdondero on 11/8/24.
 6: //
 7:
 8: import SwiftUI
 9:
10: @main
11: struct PennyIosApp: App {
12:     var body: some Scene {
13:         WindowGroup {
14:             ContentView()
15:         }
16:     }
17: }
```


blank (Page 1 of 1)

```
1: This page is intentionally blank.
```

PennyIos/ContentView.swift (Page 1 of 2)

```swift
//
//  ContentView.swift
//  Penny
//
//  Created by Katie DiPaola on 10/27/24.
//

import SwiftUI
import Combine

struct ContentView: View {

    // Note: @State means that a variable will listen for changes and
    // change within the view search results
    @State private var books: [Book] = []
    // input text
    @State var searchText: String = ""
    // timer
    @State private var cancellable: AnyCancellable?


    var body: some View {
        VStack {
            // header
            Text("Welcome to Penny IOS")
                .font(.system(size: 16))
                .padding(.vertical, 20)
            Text("Author Search")
                .font(.system(size: 32))
                .padding(.bottom, 10)
            Text("Please enter an author name:")

            // searchBar view from SearchBar.swift
            SearchBar(text: $searchText)
                .padding(.horizontal, 20)
                .padding(.bottom, 20)
                // when text is entered (aka input changes), delay and
                // then fetch from database
                .onChange(of: searchText) { newValue in
                    // Cancel the previous timer if it exists
                    cancellable?.cancel()
                    // Start a new timer for delay
                    cancellable = Just(newValue)
                        .delay(for: .milliseconds(500),
                            scheduler: RunLoop.main)
                        // fetch books from database
                        .sink { text in
                            let database = Database()
                            // handle completion (aka await return from
                            // fetchBooks)
                            database.fetchBooks(author: text) {
                                fetched_books in
                                self.books = fetched_books
                            }
                        }
                }
                // display result of current search
                NavigationView {
                    List(books, id: \.self) { book in
                        VStack (alignment: .leading){
                            Text(book.isbn + ": " + book.title)
                            Text(book.author)
                                .bold()
                        }
                    }
```

PennyIos/ContentView.swift (Page 2 of 2)

```swift
                        .background(Color(.systemGray6))
                }
            }
        }
}

#Preview {
    ContentView()
}
```

PennyIos/SearchBar.swift (Page 1 of 2)

```swift
//
//  SearchBar.swift
//  Penny
//
//  Created by Katie DiPaola on 10/27/24.
//

import SwiftUI

// re-used this code from my app
struct SearchBar: View {

    // Note: @Binding means that the source of this variable is
    // elsewhere - it's a parameter of the view

    // input text
    @Binding var text: String
    // when input is being edited
    @State private var isEditing = false

    var body: some View {
        HStack {
            // input text field
            TextField("Search ...", text: $text)
                .padding(7)
                .padding(.horizontal, 25)
                .background(Color(.systemGray6))
                .cornerRadius(8)
                // make it look pretty
                .overlay(
                    HStack {
                        // uses a build-in IOS image
                        Image(systemName: "magnifyingglass")
                            .foregroundColor(.gray)
                            .frame(minWidth: 0, maxWidth: .infinity,
                               alignment: .leading)
                            .padding(.leading, 8)

                        // provide functionality to delete full user
                        // input
                        if isEditing {
                            Button(action: {
                                self.text = ""

                            }) {
                                // uses a build-in IOS image
                                Image(systemName:
                                    "multiply.circle.fill")
                                    .foregroundColor(.gray)
                                    .padding(.trailing, 8)
                            }
                        }
                    }
                )
                .padding(.horizontal, 10)
                // track when user is editing input
                .onTapGesture {
                    self.isEditing = true
                }
        }
    }
}

struct SearchBar_Previews: PreviewProvider {
    static var previews: some View {
```

PennyIos/SearchBar.swift (Page 2 of 2)

```swift
        // view paramter (@Binding variable from above!)
        SearchBar(text: .constant(""))
    }
}
```

PennyIos/Database.swift (Page 1 of 1)

```swift
//
//  Database.swift
//  Penny
//
//  Created by Katie DiPaola on 10/27/24.
//

import Foundation

class Database {

    let url = "https://pennyjson.onrender.com/searchresults?author="

    // get books where author starts with text in author
    // completion is used for async functions to notify when the
    // function returns with data
    func fetchBooks(author: String,
        completion: @escaping ([Book]) -> ()) {
        // exit function if author is empty
        if author.isEmpty {
            completion([])
            return
        }

        // add author to database url
        let queryUrl = URL(string: url + "\(author)")!
        // make asynchronous http request
        URLSession.shared.dataTask(with: queryUrl) { data,_,error in

            if let error = error {
                print("Error fetching books: \(error)")
                completion([])
                return
            }

            // guard lets us check if an optional value exists without
            // throwing an error
            guard let data = data
            else {
                completion([])
                return
            }

            // otherwise handle json data
            do {
                // decode into list of Book objects (defined in
                // Book.swift)
                let books = try JSONDecoder().decode(
                    [Book].self, from: data)
                // completion on the main thread (UI updates must occur
                // on main thread)
                DispatchQueue.main.async {
                    completion(books)
                }
            } catch {
                print("Error decoding JSON response: \(error)")
            }
        // start task (send http request we defined above)
        }.resume()
    }
}
```

PennyIos/Book.swift (Page 1 of 1)

```swift
//
//   Book.swift
//  Penny
//
//  Created by Katie DiPaola on 10/28/24.
//

import Foundation

// book struct
struct Book: Codable, Hashable {
    let isbn: String
    let author: String
    let title: String
}
```