

PennyAndroid/activity_main.xml (Page 1 of 1)

```

1: <?xml version="1.0" encoding="utf-8"?>
2: <androidx.constraintlayout.widget.ConstraintLayout
3:     xmlns:android="http://schemas.android.com/apk/res/android"
4:     xmlns:app="http://schemas.android.com/apk/res-auto"
5:     xmlns:tools="http://schemas.android.com/tools"
6:     android:layout_width="match_parent"
7:     android:layout_height="wrap_content"
8:     tools:context=".MainActivity">
9:
10:    <LinearLayout
11:        android:layout_width="match_parent"
12:        android:layout_height="match_parent"
13:        android:orientation="vertical"
14:        tools:layout_editor_absoluteX="16dp"
15:        tools:layout_editor_absoluteY="2dp">
16:
17:        <EditText
18:            android:id="@+id/editText"
19:            android:layout_width="match_parent"
20:            android:layout_height="wrap_content"
21:            android:hint="Enter an author"
22:            android:inputType="text" />
23:
24:        <TextView
25:            android:id="@+id/textView"
26:            android:layout_width="match_parent"
27:            android:layout_height="wrap_content" />
28:    </LinearLayout>
29:
30: </androidx.constraintlayout.widget.ConstraintLayout>

```

PennyAndroid/MainActivity.java (Page 1 of 3)

```

1: //-----
2: // MainActivity.java
3: // Author: Bob Dondero
4: //-----
5:
6: package edu.Princeton.penny;
7:
8: import android.widget.EditText;
9: import android.app.Activity;
10: import android.os.Bundle;
11: import android.text.method.ScrollingMovementMethod;
12: import android.widget.TextView;
13: import android.text.TextWatcher;
14: import android.textEditable;
15:
16: import java.io.BufferedReader;
17: import java.io.IOException;
18: import java.io.InputStreamReader;
19: import java.net.URL;
20: import javax.net.ssl.HttpURLConnection;
21: import java.util.Timer;
22: import java.util.TimerTask;
23:
24: import org.json.JSONObject;
25: import org.json.JSONArray;
26:
27: public class MainActivity extends Activity
28: {
29:     private EditText editText;
30:     private TextView textView;
31:     private AuthorSearchRunnable authorSearchRunnable = null;
32:     private Timer timer = null;
33:
34: //-----
35:
36: private class AuthorSearchRunnable implements Runnable
37: {
38:     private String author;
39:     private boolean shouldStop = false;
40:
41:     AuthorSearchRunnable(String author)
42:     {
43:         this.author = author;
44:     }
45:
46:     public void setShouldStop()
47:     {
48:         shouldStop = true;
49:     }
50:
51:     public void run()
52:     {
53:         String books;
54:
55:         try
56:         {
57:             URL url = new URL("https://pennyjson.onrender.com" +
58:                               "/searchresults?author=" + author);
59:             HttpURLConnection con =
60:                 (HttpURLConnection) url.openConnection();
61:
62:             con.setRequestMethod("GET");
63:             con.setDoInput(true);
64:             con.connect();
65:             int responseCode = con.getResponseCode();

```

PennyAndroid/MainActivity.java (Page 2 of 3)

```

66:         if (responseCode != HttpURLConnection.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 savedInstanceState)
149: {
150:     super.onCreate(savedInstanceState);
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)

```

1: //
2: // ContentView.swift
3: // Penny
4: //
5: // Created by Katie DiPaola on 10/27/24.
6: //
7:
8: import SwiftUI
9: import Combine
10:
11: struct ContentView: View {
12:
13:     // Note: @State means that a variable will listen for changes and
14:     // change within the view search results
15:     @State private var books: [Book] = []
16:     // input text
17:     @State var searchText: String = ""
18:     // timer
19:     @State private var cancellable: AnyCancellable?
20:
21:
22:     var body: some View {
23:         VStack {
24:             // header
25:             Text("Welcome to Penny IOS")
26:                 .font(.system(size: 16))
27:                 .padding(.vertical, 20)
28:             Text("Author Search")
29:                 .font(.system(size: 32))
30:                 .padding(.bottom, 10)
31:             Text("Please enter an author name:")
32:
33:             // searchBar view from SearchBar.swift
34:             SearchBar(text: $searchText)
35:                 .padding(.horizontal, 20)
36:                 .padding(.bottom, 20)
37:             // when text is entered (aka input changes), delay and
38:             // then fetch from database
39:             .onChange(of: searchText) { newValue in
40:                 // Cancel the previous timer if it exists
41:                 cancellable?.cancel()
42:                 // Start a new timer for delay
43:                 cancellable = Just(newValue)
44:                     .delay(for: .milliseconds(500),
45:                           scheduler: RunLoop.main)
46:                 // fetch books from database
47:                 .sink { text in
48:                     let database = Database()
49:                     // handle completion (aka await return from
50:                     // fetchBooks)
51:                     database.fetchBooks(author: text) {
52:                         fetched_books in
53:                         self.books = fetched_books
54:                     }
55:                 }
56:             }
57:             // display result of current search
58:             NavigationView {
59:                 List(books, id: \.self) { book in
60:                     VStack (alignment: .leading){
61:                         Text(book.isbn + ": " + book.title)
62:                         Text(book.author)
63:                         .bold()
64:                     }
65:                 }

```

PennyIos/ContentView.swift (Page 2 of 2)

```

66:                 .background(Color(.systemGray6))
67:             }
68:         }
69:     }
70: }
71:
72: #Preview {
73:     ContentView()
74: }

```

PennyIos/SearchBar.swift (Page 1 of 2)

```

1: //
2: // SearchBar.swift
3: // Penny
4: //
5: // Created by Katie DiPaola on 10/27/24.
6: //
7:
8: import SwiftUI
9:
10: // re-used this code from my app
11: struct SearchBar: View {
12:
13:     // Note: @Binding means that the source of this variable is
14:     // elsewhere - it's a parameter of the view
15:
16:     // input text
17:     @Binding var text: String
18:     // when input is being edited
19:     @State private var isEditing = false
20:
21:     var body: some View {
22:         HStack {
23:             // input text field
24:             TextField("Search ...", text: $text)
25:                 .padding(7)
26:                 .padding(.horizontal, 25)
27:                 .background(Color(.systemGray6))
28:                 .cornerRadius(8)
29:                 // make it look pretty
30:                 .overlay(
31:                     HStack {
32:                         // uses a build-in IOS image
33:                         Image(systemName: "magnifyingglass")
34:                             .foregroundColor(.gray)
35:                             .frame(minWidth: 0, maxWidth: .infinity,
36:                                 alignment: .leading)
37:                             .padding(.leading, 8)
38:
39:                         // provide functionality to delete full user
40:                         // input
41:                         if isEditing {
42:                             Button(action: {
43:                                 self.text = ""
44:
45:                             }) {
46:                                 // uses a build-in IOS image
47:                                 Image(systemName:
48:                                     "multiply.circle.fill")
49:                                     .foregroundColor(.gray)
50:                                     .padding(.trailing, 8)
51:                             }
52:                         }
53:                     }
54:                 )
55:                 .padding(.horizontal, 10)
56:                 // track when user is editing input
57:                 .onTapGesture {
58:                     self.isEditing = true
59:                 }
60:         }
61:     }
62: }
63:
64: struct SearchBar_Previews: PreviewProvider {
65:     static var previews: some View {

```

PennyIos/SearchBar.swift (Page 2 of 2)

```

66:         // view paramter (@Binding variable from above!)
67:         SearchBar(text: .constant(""))
68:     }
69: }

```

PennyIos/Database.swift (Page 1 of 1)

```

1: //
2: // Database.swift
3: // Penny
4: //
5: // Created by Katie DiPaola on 10/27/24.
6: //
7:
8: import Foundation
9:
10: class Database {
11:
12:     let url = "https://pennyjson.onrender.com/searchresults?author="
13:
14:     // get books where author starts with text in author
15:     // completion is used for async functions to notify when the
16:     // function returns with data
17:     func fetchBooks(author: String,
18:         completion: @escaping ([Book]) -> ()) {
19:         // exit function if author is empty
20:         if author.isEmpty {
21:             completion([])
22:             return
23:         }
24:
25:         // add author to database url
26:         let queryUrl = URL(string: url + "\(author)")!
27:         // make asynchronous http request
28:         URLSession.shared.dataTask(with: queryUrl) { data, _, error in
29:
30:             if let error = error {
31:                 print("Error fetching books: \(error)")
32:                 completion([])
33:                 return
34:             }
35:
36:             // guard lets us check if an optional value exists without
37:             // throwing an error
38:             guard let data = data
39:             else {
40:                 completion([])
41:                 return
42:             }
43:
44:             // otherwise handle json data
45:             do {
46:                 // decode into list of Book objects (defined in
47:                 // Book.swift)
48:                 let books = try JSONDecoder().decode(
49:                     [Book].self, from: data)
50:                 // completion on the main thread (UI updates must occur
51:                 // on main thread)
52:                 DispatchQueue.main.async {
53:                     completion(books)
54:                 }
55:             } catch {
56:                 print("Error decoding JSON response: \(error)")
57:             }
58:             // start task (send http request we defined above)
59:             }.resume()
60:         }
61: }

```

PennyIos/Book.swift (Page 1 of 1)

```

1: //
2: // Book.swift
3: // Penny
4: //
5: // Created by Katie DiPaola on 10/28/24.
6: //
7:
8: import Foundation
9:
10: // book struct
11: struct Book: Codable, Hashable {
12:     let isbn: String
13:     let author: String
14:     let title: String
15: }

```