

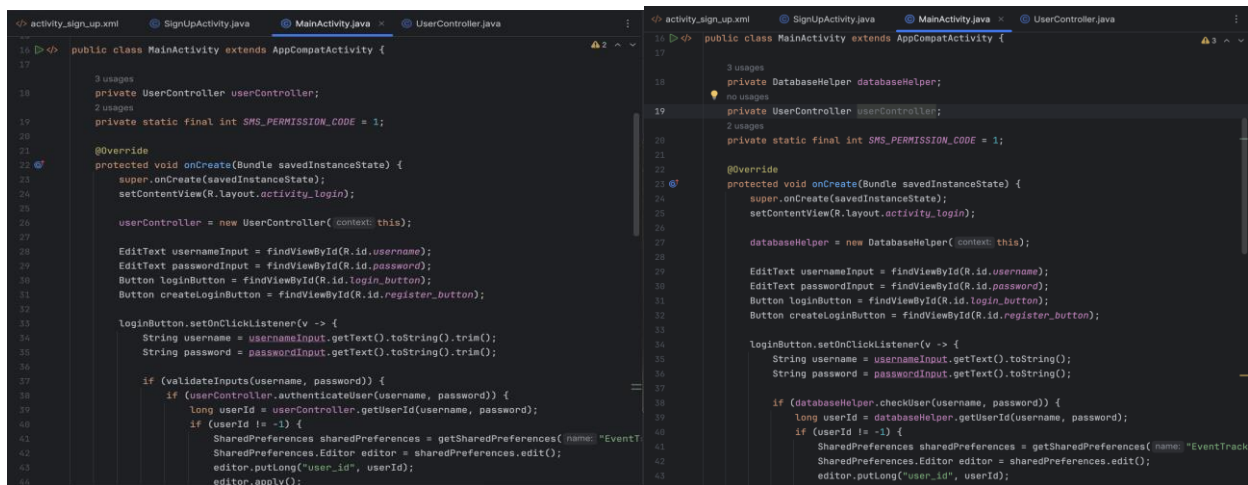


Thierry Tran

thierry.tran@snhu.edu

CS 499 Milestone Two

The Event Tracking Application is an Android-based app that I initially developed as part of my coursework in mobile programming. Its primary purpose is to allow users to create, manage, and view events with a basic user authentication system that handles registration and login functionalities. This project gave me a significant opportunity to explore mobile app development and understand how user data is managed and displayed within an app's lifecycle. I chose this artifact for my ePortfolio because it highlights my growth and progress in building practical, user-centered applications while focusing on improving software design, enhancing security, and increasing maintainability. Throughout this project, I aimed to showcase my skills in software engineering through thoughtful and meaningful improvements to the existing codebase. Specifically, the work on refactoring the MainActivity and SignUpActivity classes, as well as creating a UserController class, demonstrates how I have learned to apply the principles of modular design, improve input validation, and implement better user management. This effort was driven by a desire to showcase my ability to not only build applications from scratch but also enhance existing ones through critical improvements.

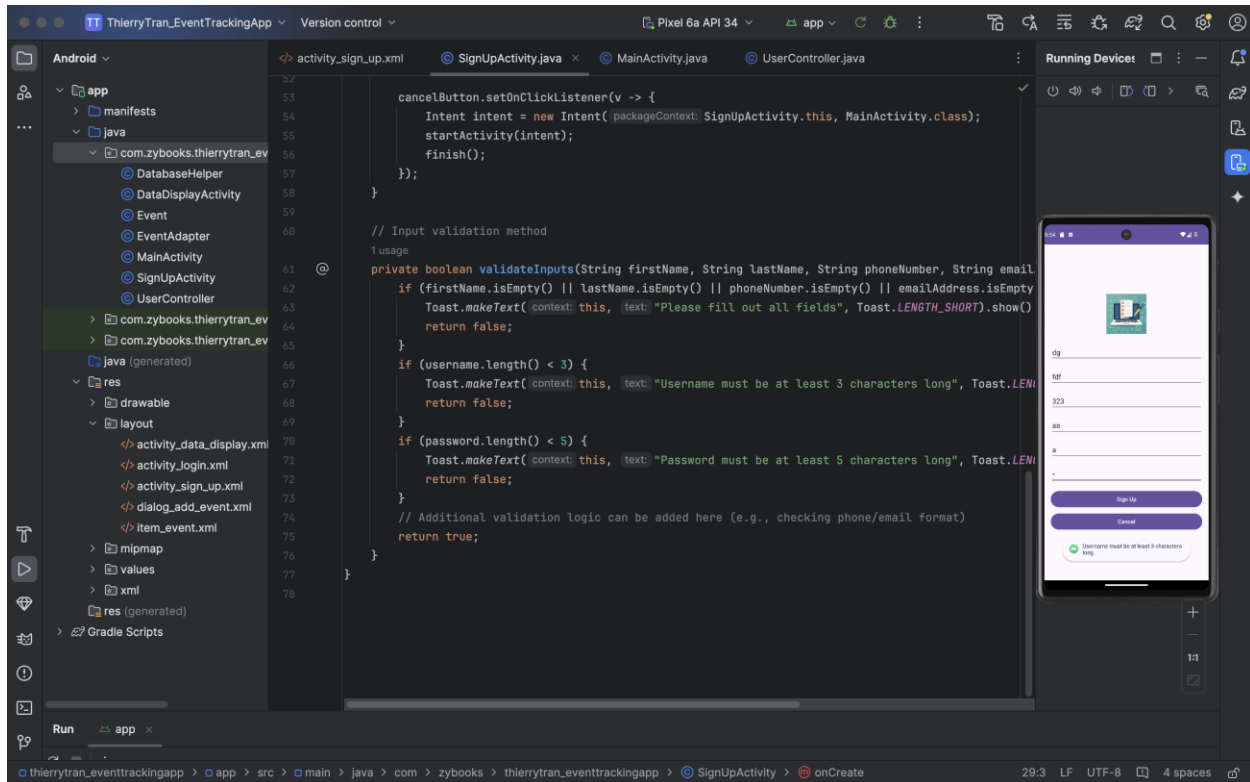


```
16  <> public class MainActivity extends AppCompatActivity {
17
18      3 usages
19      private UserController userController;
20      2 usages
21      private static final int SMS_PERMISSION_CODE = 1;
22
23      @Override
24      protected void onCreate(Bundle savedInstanceState) {
25          super.onCreate(savedInstanceState);
26          setContentView(R.layout.activity_login);
27
28          userController = new UserController(context, this);
29
30          EditText usernameInput = findViewById(R.id.username);
31          EditText passwordInput = findViewById(R.id.password);
32          Button loginButton = findViewById(R.id.login_button);
33          Button createLoginButton = findViewById(R.id.register_button);
34
35          loginButton.setOnClickListener(v -> {
36              String username = usernameInput.getText().toString().trim();
37              String password = passwordInput.getText().toString().trim();
38
39              if (validateInputs(username, password)) {
40                  if (UserController.authenticateUser(username, password)) {
41                      long userId = UserController.getUserId(username, password);
42                      if (userId != -1) {
43                          SharedPreferences sharedPreferences = getSharedPreferences("name: "EventTrack
44                          SharedPreferences.Editor editor = sharedPreferences.edit();
45                          editor.putLong("user_id", userId);
46                          editor.apply();
47                      }
48                  }
49              }
50          });
51      }
52  }
```

```
16  <> public class MainActivity extends AppCompatActivity {
17
18      3 usages
19      private DatabaseHelper databaseHelper;
20      no usages
21      private UserController userController;
22      2 usages
23      private static final int SMS_PERMISSION_CODE = 1;
24
25      @Override
26      protected void onCreate(Bundle savedInstanceState) {
27          super.onCreate(savedInstanceState);
28          setContentView(R.layout.activity_login);
29
30          databaseHelper = new DatabaseHelper(context, this);
31
32          EditText usernameInput = findViewById(R.id.username);
33          EditText passwordInput = findViewById(R.id.password);
34          Button loginButton = findViewById(R.id.login_button);
35          Button createLoginButton = findViewById(R.id.register_button);
36
37          loginButton.setOnClickListener(v -> {
38              String username = usernameInput.getText().toString();
39              String password = passwordInput.getText().toString();
40
41              if (databaseHelper.checkUser(username, password)) {
42                  long userId = databaseHelper.getUserId(username, password);
43                  if (userId != -1) {
44                      SharedPreferences sharedPreferences = getSharedPreferences("name: "EventTrack
45                      SharedPreferences.Editor editor = sharedPreferences.edit();
46                      editor.putLong("user_id", userId);
47                  }
48              }
49          });
50      }
51  }
```

The first significant change I made was refactoring the project to have a better structure. Originally, the app's logic for handling user authentication and registration was directly embedded in MainActivity and SignUpActivity, causing UI logic to be intertwined with database operations. To address this issue, I created a separate UserController class to manage user-related tasks. This change follows the Model-View-Controller (MVC) design pattern, making the application more modular, maintainable, and easier to test. This refactoring allowed me to better understand the benefits of modularization and provided a practical experience in creating maintainable and scalable codebases. By implementing this change, I also gained a deeper appreciation for the principles of separation of concerns and their impact on software design. In addition to restructuring the code, I focused on enhancing input validation to improve both user experience and security. Previously, the input fields for username and password in MainActivity and SignUpActivity had minimal validation, allowing weak or invalid credentials to be created. To strengthen this aspect, I added validation checks to enforce minimum length requirements for usernames and passwords, providing users with clear and immediate feedback when their inputs did not meet the criteria. This improvement ensures better data integrity and reduces the risk of weak passwords being used in the application. The changes were visually

represented by the updated input validation logic in the code, which effectively prevents users from creating accounts with short or empty credentials.



Security was another critical focus during the enhancement process. Before the changes, the user-related logic was exposed directly in the activities, making it difficult to manage and potentially posing security risks. Moving this logic into the UserController encapsulated the functionality and reduced potential points of failure, while also making the code more reusable and secure. The new approach highlights my understanding of secure coding practices and demonstrates my ability to create robust user authentication mechanisms.

Throughout this process, I learned that small, incremental improvements can have a substantial impact on overall software quality and user experience. While implementing these changes, I encountered challenges such as maintaining compatibility with existing functionality and

ensuring smooth integration of new logic. By carefully testing and iterating, I was able to resolve these challenges and create a more polished and user-friendly application.

Reference:

- Fowler, M. "Clean Code: A Handbook of Agile Software Craftsmanship."
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. "Design Patterns: Elements of Reusable Object-Oriented Software."